TELECOMUNICACIÓN

Campus Sur

POLITÉCNICA

ESCUELA TÉCNICA SUPERIOR DE INGENIERÍA Y SISTEMAS DE TELECOMUNICACIÓN

PROYECTO FIN DE GRADO

TÍTULO: Detección de Intrusiones basada en análisis de eventos del DNS

AUTOR: EDUARDO RAMÍREZ FERNÁNDEZ

TITULACIÓN: TELEMÁTICA

TUTOR:

FRANCISCO JAVIER ESTAIRE ESTAIRE

DEPARTAMENTO:

DTE: DEPARTAMENTO DE TELEMÁTICA Y ELECTRÓNICA

VºBº

Miembros del Tribunal Calificador:

PRESIDENTE: RAFAEL HERNÁNDEZ HEREDERO

VOCAL: FRANCISCO JAVIER ESTAIRE ESTAIRE

SECRETARIO: Mª MAGDALENA GONZÁLEZ MARTÍN

Fecha de lectura:

Calificación:

El Secretario,

A mis padres Nieves y Pepe y a Rocío,

por todo su apoyo.

"Los obstáculos no son más que un condimento del triunfo",

Mark Twain

AGRADECIMIENTOS

Han sido varios meses de duro trabajo los que me han llevado hasta este momento de culminar mi proyecto Fin de Grado. No ha sido una tarea fácil y he tenido que compaginar la elaboración de este proyecto con mi trabajo y con problemas varios, pero tal y como indica la cita al inicio del libro *"Los obstáculos no son más que un condimento del triunfo"*.

Es ahora cuando toca mirar atrás, admirar todo el camino que he recorrido a lo largo de todos estos años de carrera y agradecer a las personas que están a mi lado todo el apoyo que me han dado.

En primer lugar (ellos siempre van en primer lugar) quiero agradecerles este Proyecto Fin de Grado a mis padres Nieves y José Luis por su gran dedicación en mi educación desde que era muy pequeño y también por su gran apoyo moral, por su perseverancia, por alentarme hasta en los peores momentos. A mi hermana por darme siempre consejos tan buenos y hacerme ver que había "luz al final del túnel". También quiero agradecer el apoyo a mi novia Rocío y a mis amigos de Tarancón por estar siempre a mi lado fuera cual fuera la situación.

Gracias de todo corazón a mis amigos y compañeros de esta Escuela y sobre todo a la Asociación de Actividades Culturales, por brindarme la oportunidad de conocer un grupo de gente que me ha hecho disfrutar de la vida universitaria de una manera especial, sintiéndome parte de una gran familia ingenieril.

Especial mención a Javier, Iván, Elena, Marcos, y Luis Valero por todo el apoyo que me han dado tanto fuera como dentro de la universidad y por todos los momentos, confidencias, alegrías y penas que hemos compartido y también a Sergio, por esperar todos estos años de carrera sin que le haga una visita en Cádiz y por no cambiar en todo este tiempo. A María Dols por ayudarme y escucharme y apoyarme en todos estos años de mi estancia en Madrid.

También quería comunicar mi más sincera gratitud a Manuel Torres y a Francisco Lain, dos excompañeros de trabajo y amigos en mi estancia en Telefónica I+D, por su cercanía y por dar todo cada día en enseñarme a ser un gran profesional en mi futura trayectoria.

Por último (y no por ello menos importantes), gracias a toda mi familia (a mis tíos, a mis primos, a mi abuela, a todos) por animarme cada vez que había una reunión familiar y por sentirse orgullosos de mí desde el principio hasta el final.

Seguramente me falte alguien y pido disculpas por anticipado, pero he aprendido y he cambiado como persona por influencia de todos los que me rodean y seguramente mi actual forma de ser y parte de mis logros también hayan sido impulsados por muchos "granitos de arena" anónimos puestos en un mismo montón. A todos, gracias de todo corazón.

RESUMEN

En este proyecto se hace un análisis en profundidad de las técnicas de ataque a las redes de ordenadores conocidas como APTs (Advanced Persistent Threats), viendo cuál es el impacto que pueden llegar a tener en los equipos de una empresa y el posible robo de información y pérdida monetaria que puede llevar asociada.

Para hacer esta introspección veremos qué técnicas utilizan los atacantes para introducir el malware en la red y también cómo dicho malware escala privilegios, obtiene información privilegiada y se mantiene oculto.

Además, y cómo parte experimental de este proyecto se ha desarrollado una plataforma para la detección de malware de una red en base a las webs, URLs e IPs que visitan los nodos que la componen. Obtendremos esta visión gracias a la extracción de los logs y registros de DNS de consulta de la compañía, sobre los que realizaremos un análisis exhaustivo.

Para poder inferir correctamente qué equipos están infectados o no se ha utilizado un algoritmo de desarrollo propio inspirado en la técnica Belief Propagation ("Propagación basada en creencia") que ya ha sido usada antes por desarrolladores cómo los de los Álamos en Nuevo México (Estados Unidos) para fines similares a los que aquí se muestran.

Además, para mejorar la velocidad de inferencia y el rendimiento del sistema se propone un algoritmo adaptado a la plataforma Hadoop de Apache, por lo que se modifica el paradigma de programación habitual y se busca un nuevo paradigma conocido como MapReduce que consiste en la división de la información en conceptos clave-valor.

Por una parte, los algoritmos que existen basados en Belief Propagation para el descubrimiento de malware son propietarios y no han sido publicados completamente hasta la fecha, por otra parte, estos algoritmos aún no han sido adaptados a Hadoop ni a ningún modelo de programación distribuida aspecto que se abordará en este proyecto.

No es propósito de este proyecto desarrollar una plataforma comercial o funcionalmente completa, sino estudiar el problema de las APTs y una implementación que demuestre que la plataforma mencionada es factible de implementar.

Este proyecto abre, a su vez, un horizonte nuevo de investigación en el campo de la adaptación al modelo MapReduce de algoritmos del tipo Belief Propagation basados en la detección del malware mediante registros DNS

ABSTRACT

This project makes an in-depth investigation about problems related to APT in computer networks nowadays, seeing how much damage could they inflict on the hosts of a Company and how much monetary and information loss may they cause.

In our investigation we will find what techniques are generally applied by attackers to inject malware into networks and how this malware escalates its privileges, extracts privileged information and stays hidden.

As the main part of this Project, this paper shows how to develop and configure a platform that could detect malware from URLs and IPs visited by the hosts of the network. This information can be extracted from the logs and DNS query records of the Company, on which we will make an analysis in depth.

A self-developed algorithm inspired on Belief Propagation technique has been used to infer which hosts are infected and which are not. This technique has been used before by developers of Los Alamos Lab (New Mexico, USA) for similar purposes.

Moreover, this project proposes an algorithm adapted to Apache Hadoop Platform in order to improve the inference speed and system performance. This platform replaces the traditional coding paradigm by a new paradigm called MapReduce which splits and shares information among hosts and uses key-value tokens.

On the one hand, existing algorithms based on Belief Propagation are part of owner software and they have not been published yet because they have been patented due to the huge economic benefits they could give. On the other hand these algorithms have neither been adapted to Hadoop nor to other distributed coding paradigms. This situation turn the challenge into a complicated problem and could lead to a dramatic increase of its installation difficulty on a client corporation.

The purpose of this Project is to develop a complete and 100% functional brand platform. Herein, show a short summary of the APT problem will be presented and make an effort will be made to demonstrate the viability of an APT discovering platform.

At the same time, this project opens up new horizons of investigation about adapting Belief Propagation algorithms to the MapReduce model and about malware detection with DNS records.

ÍNDICE DE CONTENIDOS

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

LISTA DE ACRÓNIMOS

A
API→ Application Programming Interface
APT→ Advanced Persistent Threat
AXFR→ Aynchronous Full Transfer Request
B
BP→Belief Propagation
C
CDH→Cloudera Distribution of Hadoop
CIA→Central Ingelligence Agency
CLI→ Command Line Interface
CNAME→Canonical Name
CPU→Central Processing Unit
D
DNS→Domain Name System
DoS→Denial Of Service
F
FQDN→Fully Qualified Domain Name
FTP→File Transport Protocol
G
GUI→Graphical User Interface
H
HDFS→Hadoop Distributed File System
I
IO→Input/Output
IP→Internet Protocol
J
JVM→Java Virtual Machine
L
LANL→Los Alamos National Laboratory
LBP→Light Belief Propagation
M
MAC→Media Access Control
MD5→Message-Digest Algorithm 5
MEBN→Multi-Entity Bayesian Networks

N
NS→Name Server
NSA→National Security Agency
NTP→Network Time Protocol
P
PC→Personal Computer
PKI→Public Key Infrastructure
PLA→People Liberation Army
PR-OWL→Probabilistic Ontology Web Language
PTR→reverse Pointer Record
R
RAM→Random Access Memory
RHEL→Red Hat Enterprise Linux
RR→Round Robin
RRL→Response Rate Limiting
S
SCSI→Small Computer System Interface
SELINUX→Security-Enhanced Linux
SMS→Short Message Service
SOA→Start Of Authority
SQL→Structed Query Language
SSL→Secure Sockets Layer
T
TTL→Time To Live
TXT→Text
U
UML→Unified Modeling Language
URL→Uniform Resource Locator
UTC→Universal Time Coordinated
V
VPN→Virtual Private Network

2. INTRODUCCIÓN

En la actualidad nuestros equipos se encuentran conectados en red con una infinidad de dispositivos, servidores e infraestructuras que ofrecen servicios y contenidos de toda índole y origen y con infinidad de características diferentes.

Esto puede posibilitar que en ocasiones los contenidos que descarguemos puedan proceder de orígenes no del todo fiables o que, incluso, el origen del contenido o servicio utilizado sea suplantado. Todo esto puede ocasionar y ocasiona un problema en los equipos informáticos, especialmente para las empresas, que ven cómo sus datos sensibles y sus proyectos de negocio corren peligro ante la vulneración de su seguridad.

Hasta hace unos años la situación estaba suficientemente controlada. La causa era que los métodos de vulneración de la seguridad se basaban principalmente en virus organizados pero fácilmente clasificables y que diferían muy poco de casos antecedentes. Esto facilitaba enormemente la tarea de detección, ya que siempre se podían usar firmas y patrones de búsqueda. Pero la tarea se ha complicado en los últimos años a partir de la aparición de un nuevo concepto de infección, denominado "APT" (Advanced Persistent Threat).

PC World reportó (Dell Secureworks, 2012) [1] un incremento de un 81% en el número de detecciones APT sólo durante del año 2010 al 2011. Esta innovadora corriente de infección de malware es de la que se valen numerosos activistas en la red/hacktivistas (ej: Anonymous, Lulzsec…), grupos terroristas e incluso divisiones de seguridad de estados (ej: CIA, NSA, People's Liberation Army) para poder conseguir información sensible o incluso clasificada y usarla para su beneficio (ya sea este económico, político, bélico, etc…).

Es objeto del proyecto el estudio y desarrollo de soluciones para abordar este problema y buscar un nuevo enfoque a las soluciones tradicionales y así abrir una nueva vía de investigación sobre la que poder atajar las infecciones de manera más efectiva.

Este enfoque más efectivo del análisis de amenazas se deberá basar principalmente en el aprovechamiento de los sistemas de computación distribuida, la programación en modelo MapReduce y el enorme auge de los métodos de inferencia basados en modelos gráficos.

3. ANTECEDENTES Y MARCO TECNOLÓGICO

3.1. ANTECEDENTES

Podemos clasificar el malware atendiendo a sus tipos de comportamiento y a su peligrosidad observando el siguiente árbol realizado por la empresa de antivirus Kasperky (Kaspersky, 2013) [2]:

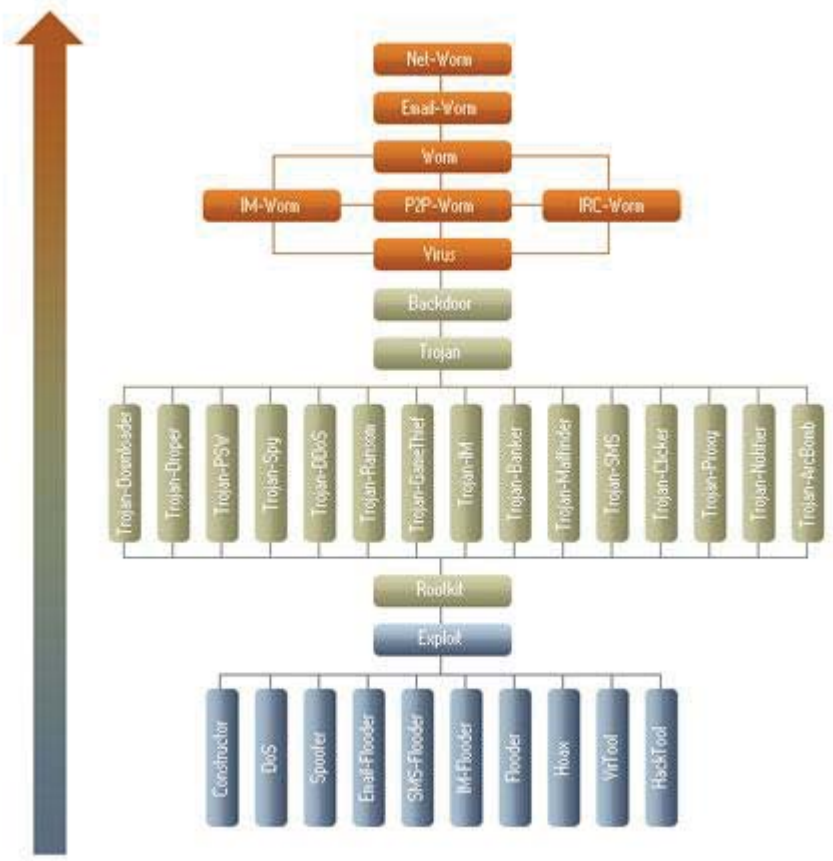

Figura 1: Clasificación por familias y peligrosidad de los tipos de malware conocidos

En la parte de abajo se observan entre las "infecciones" menos peligrosas y de comportamiento parejo: Constructor, DoS (Denial of Service), Spoofer, Email-Flooder, SMS-Flooder, Flooder, Hoax, VirTool HackTool.

- **Constructor:** Es un programa que una vez infecta al host objetivo crea nuevos virus, gusanos y troyanos. Se pueden generar desde módulos de objeto o ficheros maliciosos a nuevo código fuente.

- **DoS:** Denial of Service o Ataque de Denegación de Servicio. Inundación de los recursos de un ordenador objetivo obligándolo a procesar mayor número de solicitudes de las que es capaz.

- **Spoofer:** Software capaz de suplantar la identidad del remitente de un mensaje ya sea para ocultar el verdadero origen del mensaje o para engañar al destinatario haciéndole creer que está estableciendo comunicación con un usuario de confianza.

- **Email-Flooder:** Programa de correo basura que inunda de mensajes sin importancia las bandejas de correo electrónico.

- **Flooder genérico**: Programa que envía repetidamente mensajes a un objetivo estratégico aprovechando que sufre una debilidad o limitación (ya sea hardware o software) que puede suponer el agotamiento de recursos y el colapso del sistema.

- **SMS-Flooder**: Envío masivo de mensajes cortos de mensajería para inundar los canales con mensajes inútiles y/o saturar los sistemas de los operadores.

- **Hoax**: Son mensajes que inundan todos los canales de comunicación con la única intención de causar una alarma social ante un peligro inexistente. No son perjudiciales para los sistemas informáticos de forma directa al no contener ningún software malicioso, pero son considerados como una amenaza porque el remitente del mensaje no es legítimo y se trata de un contenido indeseado.

- **VirTool**: Software encargado de enmascarar al verdadero software malicioso para así evadir una posible detección por un Antivirus o IDS.

- **HackTool**: Alteran la lista de usuarios locales permitidos y borran logs o registros de sistema que muestren pistas de la actividad de un usuario determinado en un sistema víctima.

En un segundo grado de peligrosidad se pueden encontrar los exploits y los rootkits.

Los **exploits** son fragmentos de software, comandos o datos utilizados de forma eventual para aprovechar una vulnerabilidad en un objetivo y conseguir un comportamiento inesperado e indeseado que pueda desembocar en una vulneración completa o parcial del equipo o sistema. Pueden ser remotos, locales (necesita presencia física o posesión de una consola ya sea física o virtual con el host), o del lado del cliente (aprovecha conexiones con aplicaciones legítimas del cliente para contactar con la fuente infecciosa).

Los **rootkits** son programas que aprovechan vulnerabilidades (principalmente hardware) de un sistema para tomar control del mismo. Están implementados a bajo nivel, lo que provoca que sean invisibles para los antivirus y sistemas de detección. Además, pueden tener la capacidad de corromper a los sistemas de detección para así facilitar la entrada a nuevo software malicioso al sistema. Su eliminación puede ser complicada o imposible, especialmente si reside en el núcleo.

En el tercer grado de peligrosidad Kaspersky clasifica a los **Troyanos**. Este software malicioso, que es denominado en ocasiones Caballo de Troya se presenta como un software legítimo, de confianza y de apariencia inofensiva. En la mayoría de los casos el troyano desempeña una función útil en el sistema y tiene la apariencia de un programa de usuario, pero contiene un software malicioso que controla la máquina anfitriona sin ser advertido. El objetivo del troyano no es necesariamente destructivo o dañino.

El uso del troyano como herramienta de control de sistemas está viéndose incrementado día a día tal y como muestra la siguiente gráfica porcentual (hasta 2009) realizada por la empresa antivirus BitDefender (BitDefender, 2010) [3]:

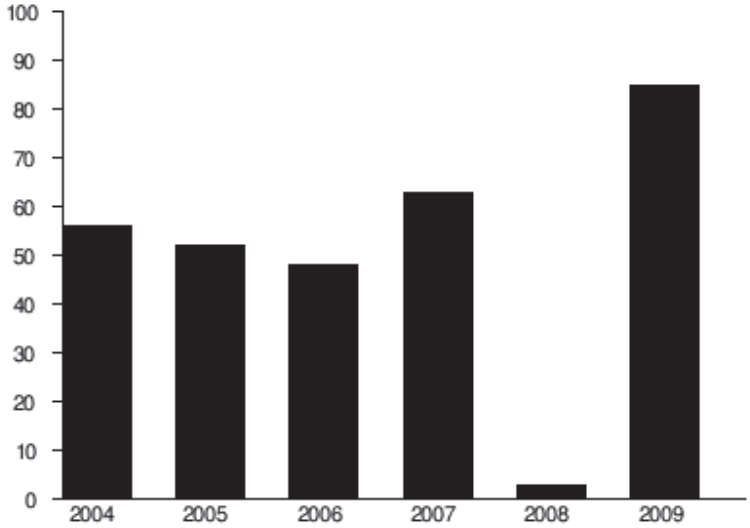

Figura 2: Evolución anual del número de troyanos detectados entre 2004 y 2009

Las **backdoors** o puertas traseras son consideradas problemas de seguridad de una severidad mayor. Consisten en métodos de accesos alternativos que esquivan la autentificación para acceder al sistema, pero algunas puertas traseras son premeditadas por los diseñadores como entrada secreta, lo que traza una "discontinua línea con ambigüedades" con el concepto de amenaza de seguridad.

En la cima de la escala de peligrosidad se encuentran los **gusanos informáticos**. Estos desarrollos software son capaces de la reutilizar procesos del sistema para hacerse invisibles y de duplicarse a sí mismo sin la ayuda de terceros. Esto provoca que un sistema que no detecte al gusano de forma inmediata puede a su vez ser infectante de otros cientos o miles de nuevos objetivos ya sea a través de SMS, mensajería instantánea, email, escritorio remoto, etc

En cambio, un APT se basa principalmente en métodos de "hacking" de alta sofisticación que requieren la existencia de un equipo infectante, un infectado y un establecimiento de comunicación de retorno ("callback") para que el infectante pueda tomar control del escenario. A pesar de lo sofisticado y elaborado que pueda estar

realizado el software de infección, para ser clasificado de APT es necesario que dicho software sea controlado por un atacante.

El nombre APT viene de los términos "Advanced", "Persistent" y "Threat". Las APTs son Avanzadas con respecto a que tienen un alto grado de sofisticación en las técnicas de infección malware. El término Persistente se refiere a que gracias a comandos ejecutados desde un equipo externo la infección y control remoto de los equipos puede llegar a durar días o incluso años si es necesario para poder obtener los datos necesarios. El concepto Amenaza hace referencia a que hay un inherente peligro debido a la intencionalidad del atacante humano que se esconde detrás

Los primeros casos reportados de uso (al menos conceptual) de las llamadas APT remontan al año 2005 en las que se usó ingeniería social para obtener información clasificada por Reino Unido y Estados Unidos.

3.2. EL INFORME MANDIANT

El informe Mandiant (Mandiant, 2013) [4] es un documento publicado en Febrero de 2013 en el que se documentan con todo lujo de detalles las actividades de ciber-guerra y ciber-terrorismo organizadas por una supuesta unidad del ejército chino (la llamada Unidad 61398 o People's Liberation Army, PLA). El documento recibe su nombre de la empresa Mandiant, que fue la que realizó la investigación Esta unidad recibe el nombre de APT1 a modo de pseudónimo y en el informe se muestran novedosos métodos y técnicas de infección usados por los atacantes, así como ingeniosas tretas de la llamada "Ingeniería social" para lograr engañar a la víctima y conseguir infiltrarse en su red.

El método de infección por APT desarrollado por APT1 (PLA) es uno de los más exitosos hasta ahora conocidos. Según un apartado del informe, PLA fue capaz de infectar, usando APTs, a al menos 141 organizaciones de los Estados Unidos y otras muchas de habla inglesa desde el año 2006 hasta 2013.

El informe, muestra no sólo datos acerca del origen de los ataques sino que realiza un análisis en profundidad en el método usado. Son desveladas y publicadas contraseñas, hashes MD5, FQDNs, certificados SSL e incluso videos de actividades capturadas en directo por los miembros de Mandiant desvelando cuál es la actividad y software malicioso ejecutado en cada máquina infectada.

Además, esta organización pseudo-delictiva seguía una estrategia bien definida que luego estudiaremos y que denominaremos: "Ciclo de vida de una APT". Los pasos más o menos eran estos:

1. Se infectaba un host de la empresa elegida con un correo electrónico suplantando la identidad de un contacto de confianza e invitando a descargar un contenido

(normalmente comprimido) que parecía finalmente estar dañado (realmente no lo estaba, era sólo un mensaje de error falso que iniciaba la ejecución del malware):

```
Date: Wed, 18 Apr 2012 06:31:41 -0700
From: Kevin Mandia <kevin.mandia@rocketmail.com>
Subject: Internal Discussion on the Press
Release

Hello,
Shall we schedule a time to meet next week?
We need to finalize the press release.
Details click here.

Kevin Mandia
```

Figura 3: Ejemplo de email de "Ingeniería social" de PLA/APT1

Se solían camuflar también ficheros ejecutables aprovechando la limitación de la longitud del nombre del fichero adjunto de los clientes de correo habituales. Así, se incluye un fichero PDF (aparentemente) con incluso el logo de Adobe Reader pero que oculta la verdadera extensión del fichero (.exe)

Figura 4: Treta de ingeniería social descubierta por Mandiant

Sorprendentemente el verdadero nombre del fichero no es "employee benefit and overhead adjustment keys.pdf" sino:

"Employee benefit and overhead adjustment keys.pdf[119_espacios].exe"

2. Se aseguraban el mantenimiento del control del host a largo plazo, permitiéndoles poder volver a conectarse. Realizaban ésto con herramientas del tipo "backdoor" o "puerta trasera", entre las que destacan las derivadas de WEBC2. Este software es un malware desarrollado por un hacker conocido como "Greenfield", que coincide con "UglyGorilla" uno de los conocidos coordinadores de U61398, la unidad militar que encubre a PLA.

3. Cubrían las comunicaciones de "callback" o llamada de retorno para gestionar y administrar los hosts infectados de forma remota. Para ello se usaban protocolos y encapsulamientos de protocolos o software conocido tales como:

Tabla 1: Protocolos mimetizados por backdoors (Mandiant)

Backdoor	Mimicked protocol
MACROMAIL	MSN Messenger
GLOOXMAIL	Jabber/XMPP
CALENDAR	Gmail Calendar

4. Obtenían mayores privilegios con programas como "cachedump, fgdump, gsecdump, lslsass, mimikatz, pass-the-hash, pwdump7 o pwdumpX". Estos programas extraen hashes, contraseñas, logins y registros de inicio de sesión de formas muy diversas.
5. Configuraban tareas programadas (at.exe de Windows) y conexiones a otras máquinas (psexec) para poder ganar más hosts infectados en la red
6. Obtenían credenciales para poder acceder con métodos más sencillos a la red (ej.: credenciales de VPN).
7. Robaban los datos que estaban buscando una vez conseguían acceso al servidor que los contiene.

3.3. CICLO DE VIDA DE UNA APT

Generalizando el caso anteriormente citado de APT1, el ciclo de vida usado en estas infecciones se basaba en el siguiente gráfico (también extraído del informe (Mandiant, 2013) [4]):

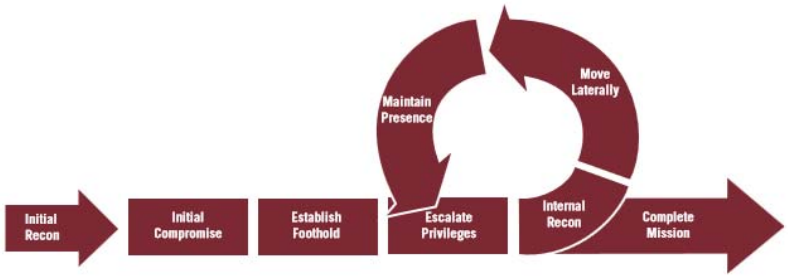

Figura 5: Gráfico del ciclo de vida de una APT (Mandiant)

Figura 6: Gráfico del ciclo de vida de una APT

1. **Compromiso inicial**: Son las tretas o mecanismos de intrusión que los atacantes utilizan para penetrar en la red de la organización objetivo. Normalmente los objetivos son individuos muy específicos dentro de la organización ya sea por su vulnerabilidad, por los privilegios que el usuario posee. Un método común de vulneración es el phising o suplantación de sitios web en la red y otro es el de introducción de código malicioso en webs de visita habitual dentro de la empresa

(normalmente por exploits) o por vulneración directa de las infraestructuras públicas de la corporación y consiguiente penetración en la red privada.

2. **Establecimiento de un punto de apoyo**: Los atacantes deben asegurarse de un agujero de seguridad permanente para realizar las intrusiones que sean pertinentes a lo largo del tiempo, ya sea por "backdoors" de fabricante, públicas o provocadas por ellos mismos. Estos "backdoors" normalmente establecen una conexión de retorno con el atacante en el proceso denominado "callback" y basado en llamadas que pueden ser muy diversas (desde texto en claro hasta comunicaciones cifradas). Ofrecen normalmente interfaces CLI o incluso interfaces gráficas para tomar control del sistema.

3. **Escalado de privilegios**: Consiste en adquirir credenciales que puedan permitir un mayor acceso a los recursos del entorno víctima. Normalmente se buscan obtener nombres de usuarios, claves, certificados PKI, clientes software VPN, ordenadores con privilegios u otros recursos que puedan permitir acceder a información de gran interés. Las APT, de hecho suelen tener como objetivo apoderarse de cuentas de administradores de dominio como primer nodo de entrada a la red, ya que esto facilita enormemente la tarea. Es común que realicen este proceso mediante la técnica "password hashes dump" que consiste en obtener passwords legítimas de cuentas de usuario a partir de romper la seguridad de los hashes de passwords almacenadas. Otra técnica usada como alternativa a la primera es la llamada "pass-the-hass", que consiste en forzar la introducción del hash en lugar de la password que lo genera (con lo que no es necesario el proceso de romper el algoritmo de hash). Hay numerosas herramientas en la red que pueden realizar este tipo de tareas.

4. **Reconocimiento interno**: Este proceso se basa en el reconocimiento de la distribución corporativa y recolección de información del entorno objetivo. Normalmente se usan herramientas básicas de red basadas en inspecciones básicas de tablas ARP y otras más complejas para adivinar cuál es la topología de la red y otros datos como los usuarios, grupos y relaciones de confianza. Los objetivos principales son documentos, contenidos de cuentas de email de los usuarios, bases de datos… Normalmente se realiza mediante scripts automáticos creados por el atacante.

5. **Movimiento lateral**: En muchas ocasiones los sistemas originalmente vulnerados por los intrusos no contienen los datos que ellos desean, por lo que necesitan moverse lateralmente a lo largo de la red para poder encontrar que equipos contienen los datos y acceder a ellos. Este proceso supone que más equipos deberán ser vulnerados e infectados con software APT y se usa la herramienta "crontab" o "programador de tareas" del equipo para permitir la ejecución periódica de software malicioso controlado a remoto.

6. **Mantener la presencia:** Se deben tomar acciones para garantizar que el control de los equipos del entorno es continuado desde el exterior de la red. Esto se garantiza mediante la instalación de nuevos "backdoors" durante el desarrollo de la operación para evitar que se corte el acceso si se localizan los anteriores. Para

complicar aún más la tarea al equipo de seguridad de la empresa infectada, normalmente se suelen usar múltiples direcciones remotas desde las que se controlan y estas van mutando periódicamente. Además se usa un conjunto de software muy variado y cambiante distribuido por un conjunto grande de hosts infectados en la red. También se suelen abrir otros métodos de acceso alternativos a los tradicionales "backdoors" (PKI, credenciales VPN, enmascaramiento de usuarios ilegítimos por legítimos, doble camino de autenticación…).

7. **Completar la misión:** Una vez robados los ficheros de interés en los sistemas comprometidos se procede a empaquetarlos o comprimirlos en otros archivos para ocultarlos y finalmente robalos. Normalmente se usa el formato .RAR para realizar esta tarea, aunque también se han visto casos en los que se usaban formatos .ZIP o .7-ZIP. Es común que se minimice el uso de la compresión pero sí se protejan los datos con una contraseña. Para extraer los datos de la red se buscan soluciones muy dispares dependiendo de la seguridad, topología del entorno vulnerado, etc. Entre los métodos más habituales se encuentran el protocolo FTP (File Transfer Protocol), "backdoors" ya existentes o métodos personalizados de transferencia de archivos.

3.4. EL PROTOCOLO DNS

El protocolo DNS es usado para poder traducir conceptos y direcciones comunes usadas por humanos a direcciones IPs (usadas por los hosts de una red). Para completar una resolución DNS es necesario completar una serie de pasos, pero hay dos tipos de consultas: iterativa y recursiva.

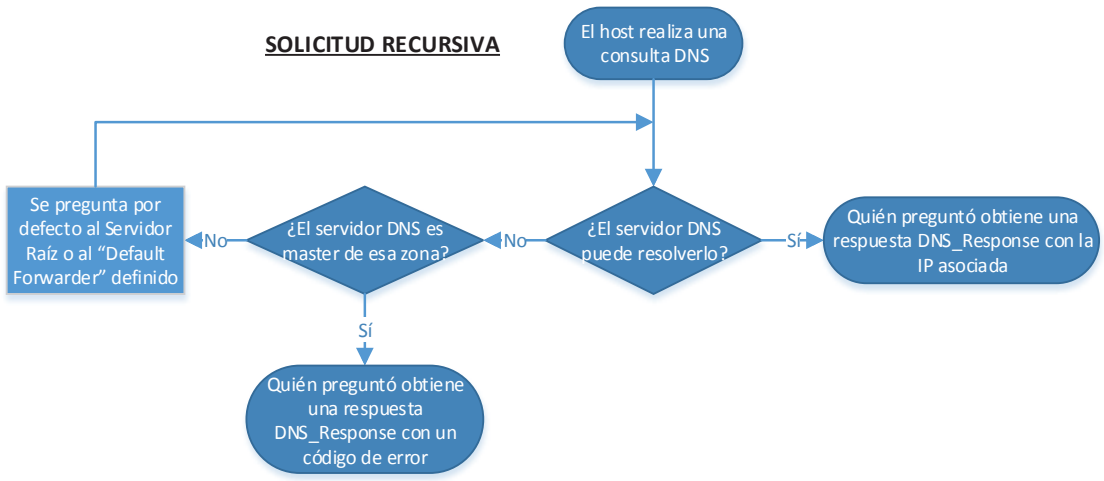

Figura 7: Diagrama de secuencia del protocolo DNS en modo recursivo

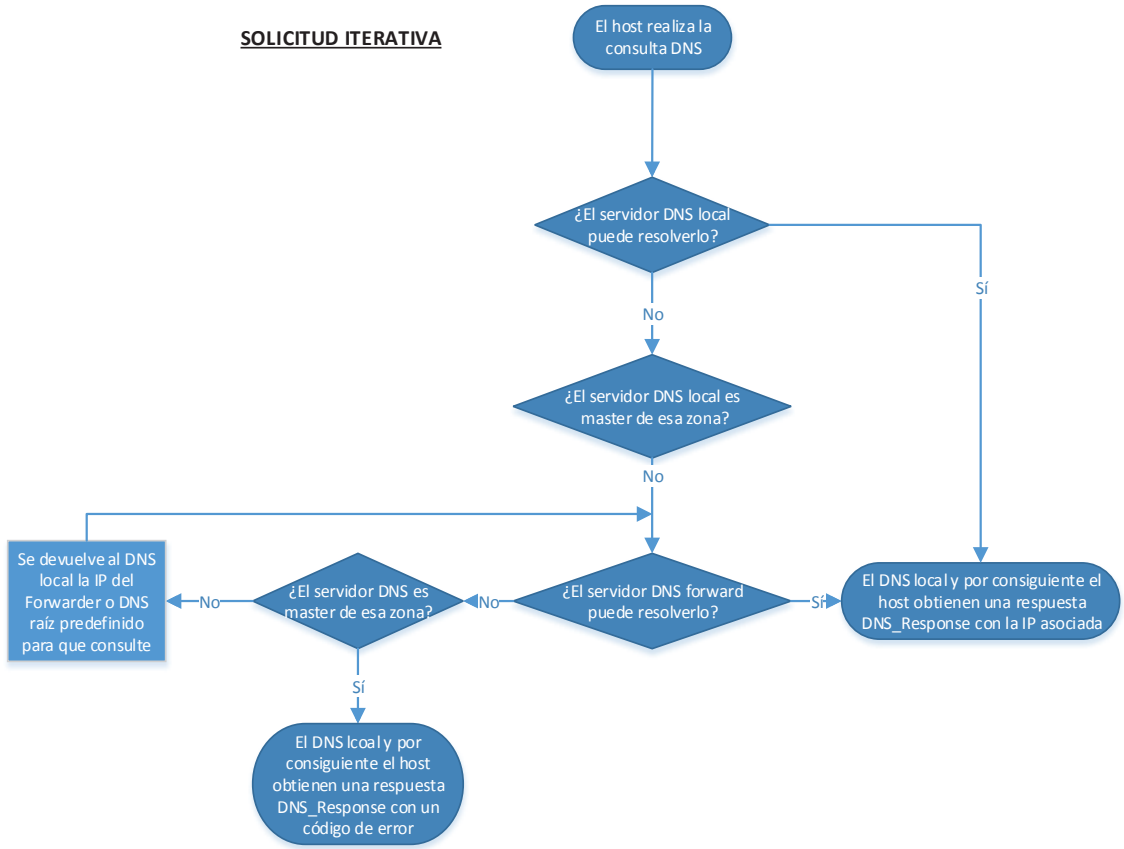

Figura 8: Diagrama de secuencia del protocolo DNS en modo iterativo

Secuenciación de los mensajes para un ejemplo de solicitudes recursivas (Roolvink, 2008)[5]:

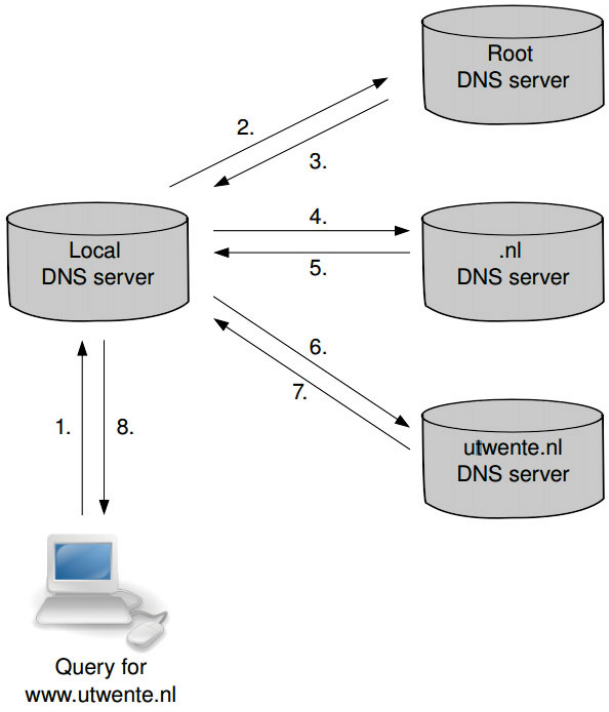

Figura 9: Diagrama de secuencia numerado de una solicitud recursiva

Secuenciación de los mensajes para un ejemplo de solicitudes iterativas[5]:

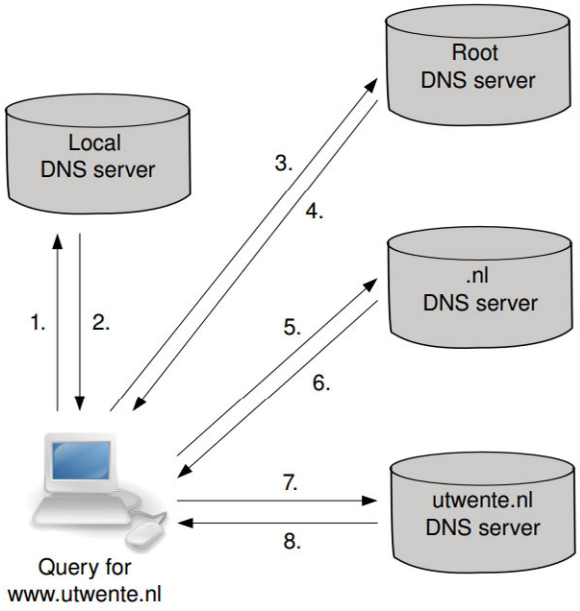

Figura 10: Diagrama de secuencia numerado de una solicitud iterativa

Tipos de registros DNS más frecuentes:

- Registro A → Registro de direcciones. Mapea hostnames a una IPv4.
- Registro AAAA → Registro de direcciones v6. Mapea hostnames a una IPv6.
- Registro CNAME → Registro de alias. Mapea un alias a otro hostname y sigue mapeando hasta alcanzar la IP asociada.
- Registro MX → Mapea un nombre de dominio a una lista de MTA (correo) para ese dominio
- Registro NS → Sirve para delegar una determinada zona a un determinado servidor DNS
- Registro PTR → Indica cuál es la resolución inversa para un registro (IP → nombre de host)
- Registro SOA → Especifica los datos autoritativos de una zona DNS determinada (DNS primario, email del administrador de dominio, número de serie, tiempos de actualización, etc.
- Registro TXT → Indica comentarios legibles por seres humanos, pero también puede incluir parámetros para procesos automáticos de algunos servicios.
- Registro AXFR → Consiste en transferir toda la zona del servidor maestro a los esclavos

Hay múltiples registros DNS además de estos aquí mencionados. Si se quiere tener una lista completa se puede ver una lista completa en la RFC-1035 (IETF, 1987) [6]

Tipos de respuestas DNS:

- NOERROR = 0 → La respuesta se ha producido sin problemas
- FORMERR = 1 → Error de formato DNS, la solicitud no cumple la especificación ya que contiene errores en algunos campos.
- SERVFAIL = 2 → El DNS presenta un fallo en el servicio. Puede deberse a múltiples causas tales como una mala configuración, una suspensión de servicio debido a un ataque DoS (Denial Of Service), el estado de la zona es corrupto, etc
- NXDOMAIN = 3 → No existe la entrada que estás buscando en este dominio. El dominio existe pero no la entrada solicitada en el servidor
- NOTIMP = 4 → El servidor de DNS no soporta el código de operación especificado, la función no está implementada pero la solicitud es correcta.
- REFUSED = 5 → El servidor DNS objetivo se está negando a dar respuesta o actualización de la entrada solicitada. Se ha rechazado la solicitud de forma explícita debido a que no cumple las políticas establecidas, a pesar de que sí que cumple el formato de la especificación. Un ejemplo es si el origen de la petición es considerado como "no deseado" o si el DNS está configurado para servir a direccionamiento privado y la solicitud es realizada desde direccionamiento público.
- YXDOMAIN = 6 → El nombre existe pero no debería existir (poco común)
- YXRRSET = 7 → Un registro que no debería existir existe. Estas intentando actualizar una entrada ya existente.
- NXRRSET = 8 → Un conjunto que debería existir pero no existe (poco común)
- NOTAUTH = 9 → El DNS señalado no es autoritativo para la zona nombrada en la sección (estás intentando actualizar un registro de zona en un servidor que no corresponde)
- NOTZONE = 10 → Un nombre usado en las secciones de "Prerrequisito" o "Actualización" no pertenece a la zona especificada

3.5. LOGS DE DNS

Normalmente en un servidor DNS todas las consultas hechas por clientes quedan referenciadas en un histórico. En dicho histórico se hace referencia a todos los factores citados anteriormente: IP/nombre origen, tipo de consulta realizada, consulta realizada, IP del servidor y además una hora de consulta o timestamp.

En el caso de un servidor DNS troncal de una empresa, y más si este sirve como "último eslavón" al contacto con los servidores root y la salida a Internet, este histórico (comúnmente denominado "log") puede servir para realizar una estimación muy aproximada de que páginas web y hosts externos se han visitado. El log tiene una apariencia muy similar a esta:

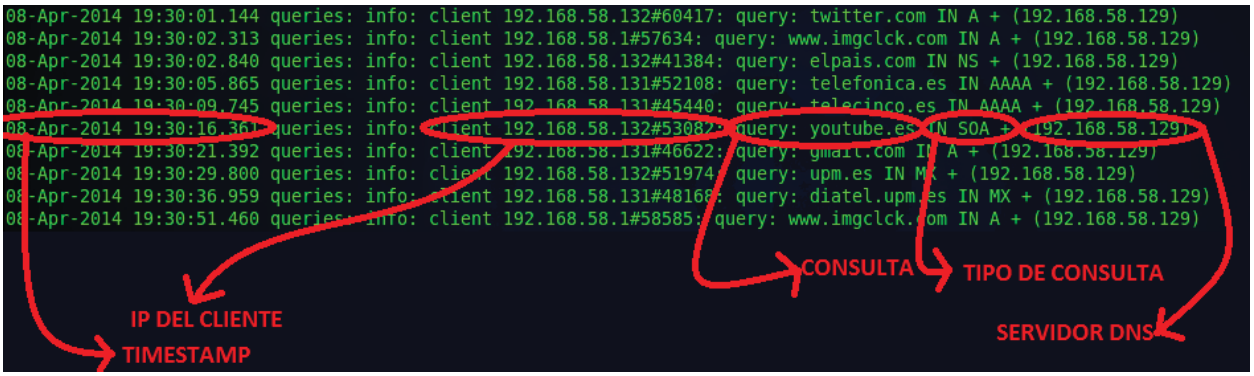

Figura 11: Diferentes datos extraíbles de un log de DNS

3.6. ATAQUES Y VULNERABILIDADES DNS

- Envenenamiento DNS (DNS cache poisoning)

Consiste en que el atacante explota una vulnerabilidad inherente en el protocolo DNS y el servidor vulnerado acepta información incorrecta que ha sido alterada con fines maliciosos. Al no comprobar si los mensajes provienen de una fuente autoritativa, el servidor almacena de forma local información incorrecta y la envía los usuarios a los que está dando servicio. El resultado es que el contenido de resoluciones DNS resulta siendo borrado o alterado, redirigiendo a los clientes a webs diferentes a las originales y estas pueden contener contenido malicioso, phising (sustituyendo las resoluciones de webs bancarias por IPs de hosts con webs que suplantan la identidad de ese banco, por ejemplo)…

Una variante de este ataque es la de redirigir todas los dominios de los que un servidor es master/autoritativo, a un nuevo servidor DNS que es el que se encarga de dar la resolución alterada definitiva. Esta técnica es mucho más sencilla, ya que permite la redirección del tráfico sin tener que editar registro a registro. El problema que presenta es que también es más difícil que tenga éxito (no siempre podemos reencaminar el tráfico

DNS a otros servidores ya que puede haber elementos de red que nos lo impidan) y es más fácilmente detectable (sospechosamente todo el tráfico DNS se redirige a una única IP).

Otra variante es dar la respuesta a la consulta realizada antes de que el servidor DNS auténtico la pueda dar, es lo que se llama "falsificación DNS" (DNS Forgery). Este ataque cada vez está más limitado ya que los servidores DNS escogen un puerto al azar en cada respuesta que dan y no por cada vez que arrancan. Toda esta información se puede consultar con mayor detalle en (HoneyNet, 2007) [7]

- Fast flux

Es una técnica que usa el DNS para ocultar la IP de orígenes maliciosos. Normalmente se montan redes de equipos comprometidos botnets a los que apuntan los registros DNS de un dominio determinado. Estas botnets sirven de proxy entre los clientes y los servidores donde se almacena el contenido infectado.

Los hosts proveen un enorme rango de direcciones a una entrada DNS concreta, teniendo así un Round Robin que alterna entre todos los equipos infectados botnet como dirección de registro (para esto tienen un TTL muy bajo).

Este engaño puede ser a uno o dos niveles (Single/Double-Flux) dependiendo de si el DNS que atiende las peticiones (registro NS) es uno diferente al de la dirección vulnerada o no.

Mediante este tipo de engaño los atacantes consiguen una estructura sencilla con la que poder obtener datos o infectar usuarios ya que tan sólo necesitan vulnerar el DNS y se ahorran una gran cantidad de coste en mantenimiento y disminuyen los riesgos (los hosts que hacen RR son botnets que no tienen por qué pertenecerles). Además se tiene la ventaja de que los hosts atacantes realmente son hosts infectados repartidos por multitudes de países y esto dificulta posibles investigaciones o complicaciones con las autoridades. También es mayor el nivel de infección (pueden llegar a formarse botnets de miles de hosts), la duración de la suplantación (se alarga el proceso de detección e identificación). Se pueden observar ejemplos prácticos en (HoneyNet, 2007) [7], de donde también se han extraído las siguientes imágenes:

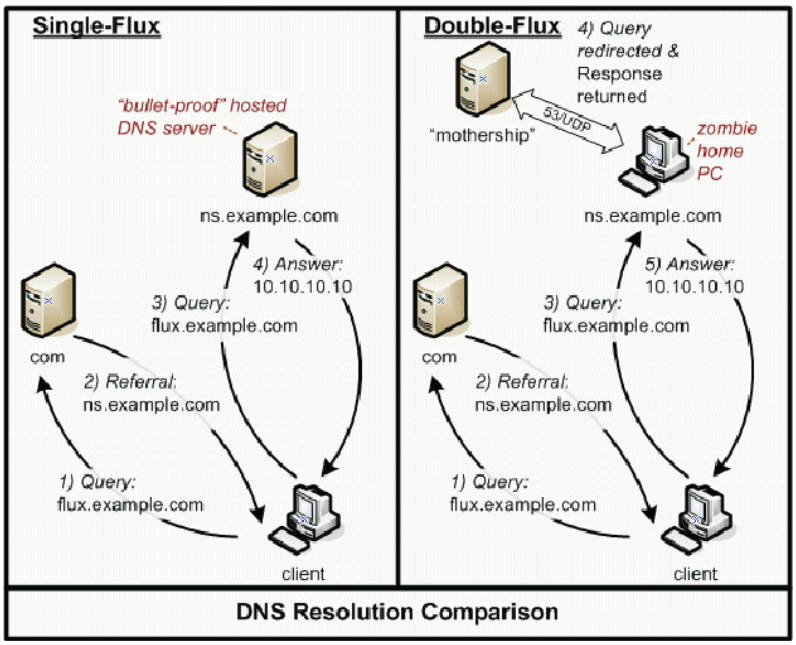

Figura 12: Comparación del ataque Single/double Flux con respecto a solución DNS

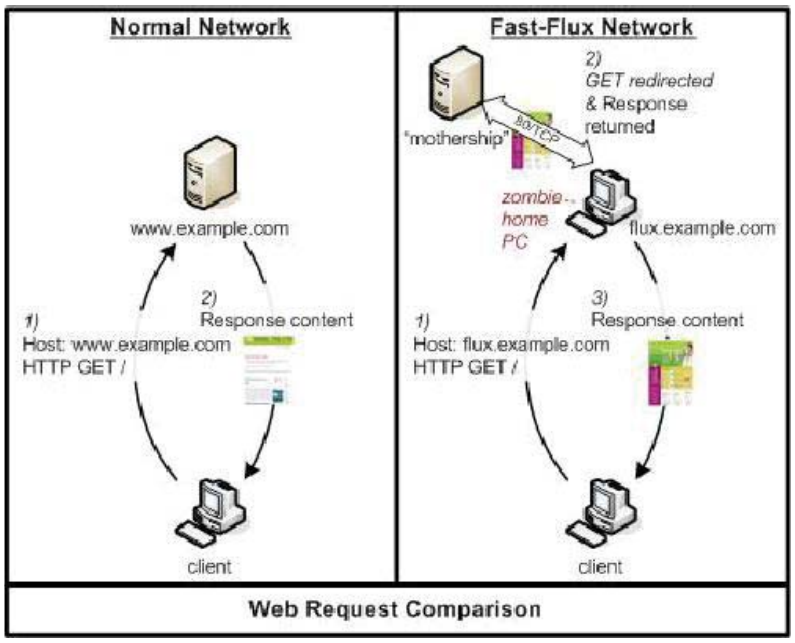

Figura 13: Diferencia de resultados web con una red infectada con técnicas Fast-Flux

- Ataque de amplificación DNS

Es el nombre por el que es conocido un tipo de ataque de denegación de servicio (DDoS) que se basa en el uso de servidores DNS de acceso público para sobrecargar a la víctima con una inundación de respuestas de dicho servidor o servidores.

Consiste en que el atacante hace una consulta al servidor DNS que servirá de amplificador, pero se modifica el paquete (spoofing) para que la IP del solicitante sea otra diferente a la IP del atacante. Cuando el servidor DNS envía la respuesta la enviará entonces a la víctima, sirviendo de "proxy" y cumpliendo una doble función:

- El servidor DNS oculta (al menos parcialmente) la identidad del atacante
- Las respuestas del servidor DNS son de mayor tamaño que la consulta (de hecho el atacante suele hacer una consulta del tipo ANY, que devuelve toda la información sobre la zona DNS es una sola respuesta), lo que amplifica (de ahí el nombre) el efecto del ataque.

Este tipo de ataque se suele basar en el uso de botnets y redes de hosts zombies infectados que hacen consultas masivas infectadas haciendo "spoofing" de la IP de origen. Esto desencadena en que la inundación de la víctima se puede llegar a ser masiva con un esfuerzo muy pequeño.

Para mitigar este tipo de ataques se usa la técnica Response Rate Limit, en la que se fija un contador máximo de consultas por unidad de tiempo, un tiempo de "baneo" de consultas y una máscara de red a la que se aplica. Por ejemplo: "Si se reciben más de 20 consultas por segundo de una red de máscara /24, bloquea las respuestas durante 10 segundos a toda la subred". Una buena configuración del parámetro RRL debe permitir bloquear no sólo al host que excede la tasa sino a toda su posible red botnet asociada.

3.6. DETECCIÓN DE APTS

Actualmente los métodos de descubrimiento de APTs se basan en el uso de firmas, reglas aplicadas de forma estática sobre el tráfico de red, y análisis forense del tráfico (normalmente a partir de técnicas que detectan diferentes comportamientos sospechosos: MAC Spoofing, Rogue DHCP, MAC Spoofing, DHCP ACK Injection,..), así como técnicas de geolocalización… Esta situación se complica cada vez más debido a la diversidad creciente del software APT, pero los métodos tradicionales siguen suponiendo en un gran porcentaje de los casos una garantía de detección suficientemente alta si el "secreto" guardado no es muy valioso.

El ARP/MAC Spoofing se basa en la utilización de una dirección MAC (física) copiada para poder acceder a ciertos recursos de red y suplantar la identidad de un nodo concreto de una red. El método detección más extendido es el de la detección pasiva. Se hacen tablas permanentes de asociaciones de pares IP-MAC. Si se detecta un cambio en una asociación IP-MAC se avisa con una alerta y se notifica así de una posible suplantación. Una conocida herramienta de detección MAC para entornos Linux/Unix es *arpwatch*.

El Roque DHCP consiste en que un servidor DHCP intruso entra en nuestra red para suplantar a nuestro DHCP actual. Con esto se consiguen que algunos parámetros (encaminamiento, DNS, etc) sean modificados con facilidad por el servidor impostor. El método de detección se basa en mecanismos de escucha en la red para detectar IPs duplicadas (pasivo) o de lanzamiento masivo de solicitudes DHCP hasta la detección de un servidor DHCP que ofrezca datos de red (DNS, gateway…) diferentes a lo esperado.

DHCP ACK injection consiste en un ataque similar al Rogue, pero algo más elaborado (no se puede descubrir de forma pasiva). Aun así la detección de ataques basados en DHCP es altamente ineficiente (y cada vez más) debido a que aparecen muchos falsos positivos. Los falsos positivos son detecciones realizadas por los mecanismos anti-malware ante eventos que no presentan ningún tipo de amenaza. Un ejemplo de esto para DHCP puede ser el caso de los “DHCP Failover”, en los que se presentan 2 DHCPs que trabajan en el mismo rango de direcciones y contestan al mismo tiempo; este caso podría ser un falso positivo.

Una herramienta muy comúnmente usada para la detección APT actualmente es la herramienta de preprocesamiento Snort. Snort es un sistema IDS basado en firmas (con un motor de reglas basadas en patrones) que puede alertar sobre posibles anomalías o ataques en nuestra red. Este tipo de sistemas tiene la limitación de que necesita conocer el ataque (base de datos/firmas) para poder detectarlo, lo que supone una desventaja en entornos de constante cambio y amenaza. Se puede buscar más información acerca del tema, incluyendo ejemplos prácticos en (INTECO, 2013) [8].

Ante esta situación se necesita un inminente cambio para poder garantizar que la seguridad es lo suficientemente dinámica, flexible y adaptativa como para poder resolver los problemas de una empresa. Además la posible solución propuesta debe ofertar una alternativa más barata a los productos ya existentes.

Es por este motivo por el que en este proyecto se plantea que hay una necesidad imperante de encontrar un punto en común en todos los malware APT e iniciar un proceso de investigación y detección a partir de dicho punto. En este caso el punto escogido ha sido los eventos DNS por su sencillez y cercanía al usuario final y porque la mayoría de las empresas tienen un servidor DNS propio y tienen los logs del servicio a su disposición.

4. PLATAFORMA DE DETECCIÓN DE APTS

4.1. DETECCIÓN DE APTS A PARTIR DE LOGS DE DNS

Ahora que conocemos lo que es una APT, el protocolo DNS y cómo funciona se puede ya tener una idea de cuál es el motivo por el que utilizar los logs de DNS para descubrir posibles infecciones en una red corporativa.

En primer lugar, las APTs, tal y cómo hemos visto antes son amenazas persistentes a lo largo del tiempo y necesitan contactar mediante una llamada al origen de infección. Debido a posibles contramedidas de seguridad el origen de la infección o "control de mando" no puede ser una IP estática a lo largo del tiempo ni un nombre estático. Esto lleva a que las APTs cambian de IP y de nombre de "callback" periódicamente para complicar la tarea a los equipos de seguridad de las empresas.

Además, las APTs también son capaces de camuflarse perfectamente en un entorno empresarial usando puertos de uso común (como por ejemplo puede ser el de videollamada, el de escritorio remoto, etc) para no levantar sospechas y además pueden "esnifar" la red para detectar las horas puntas de tráfico y así no parecer sospechosas enviando paquetes a horas fuera del horario laboral, por ejemplo.

Ante este hostil y difícil escenario el viejo paradigma de aprendizaje de un antivirus queda totalmente obsoleto ya que los patrones mutan constantemente, los orígenes y destinos también, así como las horas y los métodos… Es el momento en el que el DNS puede ser la pista de indicio para desenredar esta maraña de flujos y discernir entre los que son legítimos y cuáles pueden provenir de una máquina infectada o vulnerada.

Normalmente existe un gran número de pares DNS-IP sobre el que los atacantes tienen control. Y se podría estudiar este fenómeno de descubrimiento de APTs desde el ámbito de los datos de registro (WHOIS) o la asociación nombre-IP (en este caso se ha elegido el segundo estudio).

El estudio de redes Fast-Flux (muy utilizadas para esconder bajo una segunda capa de redirección, a los originales atacantes), por ejemplo, demuestra que los cambios en este tipo de infraestructuras orientadas a malware originan información WHOIS mucho más cambiante que las de redes normales. Esto nos llevaría a pensar que estudiando la velocidad y prolongación en el tiempo con el que percibimos cambios en el WHOIS de un dominio sospechoso nos podrá revelar un mayor número de información sobre los atacantes. Pero este no es el objetivo de este proyecto de investigación, sino el de observar la influencia de las APTs en relación al par nombre-IP.

En relación con todo lo anteriormente citado se parten de los siguientes axiomas:

- Toda máquina (incluso las infectadas) necesitan contactar con el DNS para poder conocer la IP del website, servicio o destino al que quieren conectarse.
- El DNS es capaz de registrar en el log todas las consultas y decirnos con el timestamp horas muy precisas a las que se realizaron consultas. Esto nos puede dar información de coincidencia o periodicidades en tiempos de consulta y darnos pista de procesos "no humanos" en equipos de usuario. Como ejemplo el servidor DNS bind9 da una exactitud de milésimas de segundo en el log:

```
08-Apr-2014 19:32:42.458
```

Figura 14: Ejemplo de timestamp de un log DNS

- De las APTs se conoce siempre algo: están obligadas a cambiar su dirección de "callback" y se podrá aprovechar esto, ya que en su mayoría deben tener un conjunto limitado de direcciones para que los dos extremos se lleguen a sincronizar. Esto significa que muy probablemente se puedan encontrar patrones periódicos de consulta a lo largo de un tiempo suficientemente grande. También significa que la máquina infectada está sujeta a realizar esporádicas consultas o resoluciones erróneas intentando llamar a una dirección del "pool" DNS de "callback". Se puede consultar más información en: (NASA, LANL, 2014) [9]

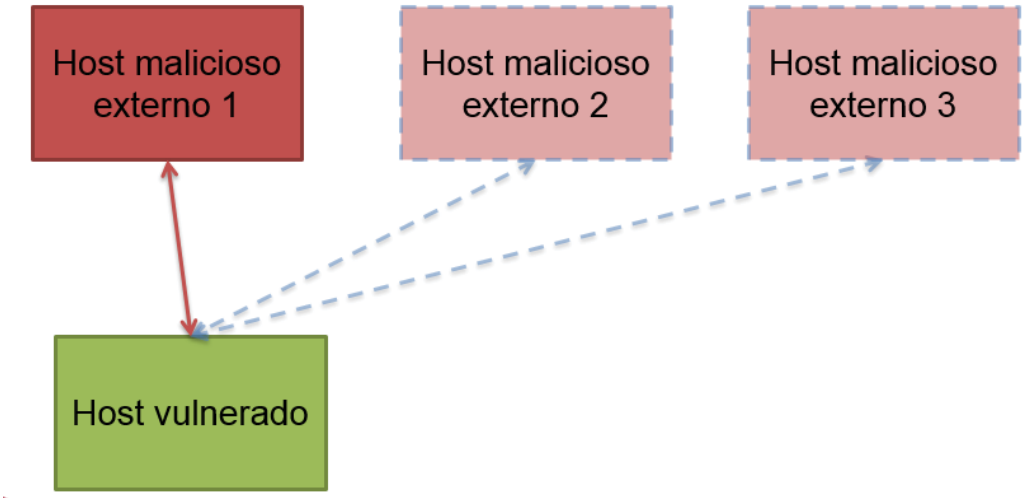

Figura 15: Ejemplo de APT en proceso de "callback" con varios hosts externos

- Se debe tener en cuenta que el DNS genera una enorme cantidad de información de log. Si se revisan logs previos a la infección se puede contrastar cuáles son las direcciones externas novedosas (podría sorprender la improbabilidad de encontrar una dirección web nueva en una gran empresa tras el almacenamiento de un par de semanas de log de DNS). Esto reduce la búsqueda de manera sustancial.

- Se tiene certeza sobre que las APT tienen una fase de movimiento lateral, por lo que una vez fijados nodos sospechosos en la red se incrementará notablemente su posibilidad de ser maliciosos si otros hosts con los que han contactado (flujo del que probablemente también se puede tener constancia en DNS) también atacan a un conjunto similar o parecido de dominios maliciosos, realizan comportamientos periódicos extraños y/o tienen errores en consultas en el mismo dominio o relacionados.

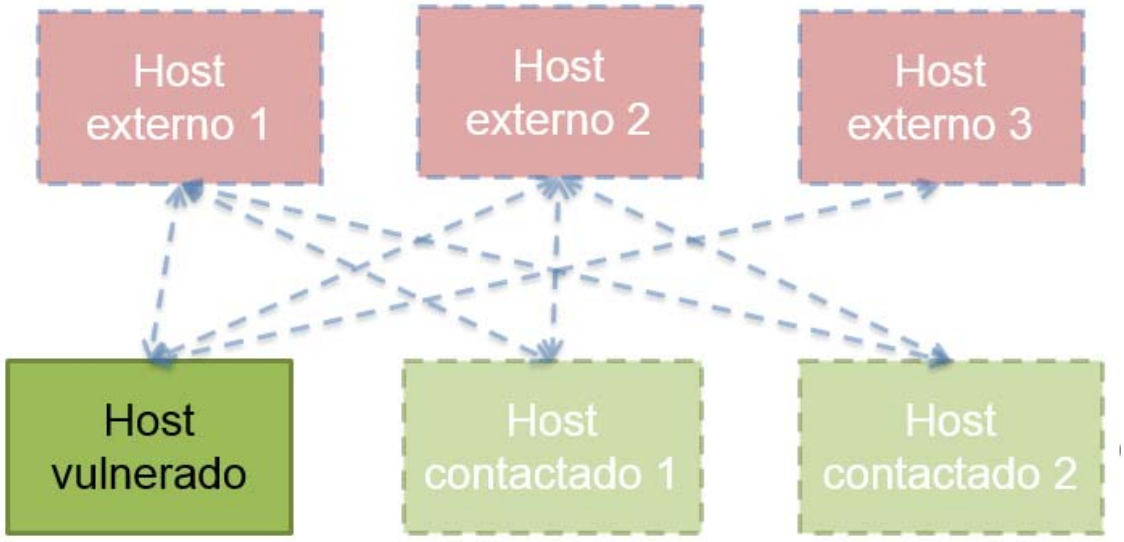

Figura 16: Varios posibles hosts infectados por la APT en "callback"

4.2. DESAFÍO "APT INFECTION DISCOVERY USING DNS DATA"

El 30 de Abril de 2013 el laboratorio de Los Álamos (Nuevo México, EEUU) del Gobierno de los Estados Unidos lanza un desafío al mundo entero (LA-UR-13-23109, (NASA, LANL, 2013))[10]. La intención es la de avanzar en el desarrollo de una plataforma software capaz de descubrir los hosts de una red que han sido infectados con malware del tipo APT usando como única base de indicio los logs de DNS.

Para conseguir este objetivo del LANL, el gobierno de los Estados Unidos libera unos ficheros con gigantescas cantidades de logs DNS (unos 28GB por día) que contienen la actividad de una red corporativa durante el periodo de 16 días. Esto supone una cantidad de datos de unos: 28GBx16=448GB.

Debido a la gran cantidad de información esta fuente de información ofrece y a que ya contienen en su interior información relacionada con hosts infectados por herramientas APT, se ha decidido usar estos logs como inicio para realizar la tarea de descubrimiento.

Pero para poder realizar la tarea correctamente se deberá tener en cuenta que el tamaño de estos logs requerirá especificaciones hardware y de plataforma muy especiales y que el formato de estos logs no es el "estándar".

Los logs han sido tratados previamente con una herramienta que modifica el formato de los mismos para hacerlos más intuitivos y fáciles de leer. Para su formateo se ha usado la herramienta llamada dns_parse, también desarrollada por Los Álamos y que puede ser descargada libremente de esta dirección: https://github.com/pflarr/dns_parse

Así, el formato de los logs DNS pasa a ser como el del ejemplo que ahora se muestra:

Query:

```
<SEP>
2013-03-12 00:01:05.995883,74.92.139.33,74.92.185.4,X,u,q,NA
? overpower.webb.bin A
```

Figura 17: Ejemplo de consulta de DNS formateada por dns_parse

Response:

```
<SEP>
2013-03-12 00:01:05.996197,74.92.185.4,74.92.139.33,X,u,r,AA
? overpower.webb.bin A
! overpower.webb.bin CNAME disenfranchising.webb.bin
! disenfranchising.webb.bin A 74.92.98.3
```

Figura 18: Ejemplo de respuesta DNS reformateada con dns_parse

Se puede observar que las consultas y respuestas tienen un formato basado en:

<Línea de separación>

[Fecha] [Hora],[IP origen], [IPdestino], [Etiqueta]

[Símbolo] Consulta/Respuesta

…

<Línea de separación>

La "Etiqueta" consiste en 4 grupos de letras separados por comas que indican que tipo de interacción DNS se está realizando. Para el caso que nos atañe solo se encuentran dos tipos de etiquetas: X,u,q,NA (consulta), X,u,r,AA (respuesta).

El "Símbolo" consiste en un carácter informativo para diferenciar qué parte del mensaje DNS fue solicitada por el cliente y qué parte fue respondida/añadida por el servidor. Así los 3 principales símbolos que se pueden encontrar son: "?"(preguntado por

el cliente", "!"(respondido por el servidor), "$"(redirigido a otro servidor/forward), "+" (opciones adicionales).

Además, en el desafío del laboratorio de Los Álamos se especifican 4 casos para poder afrontar el desafío de manera progresiva, que aunque en este proyecto no se van a seguir (el problema no está abordado en tanta profundidad) son los siguientes:

- Caso 1: Encuentra los dominios maliciosos
- Caso 2: Encuentra el ataque conociendo un conjunto de hosts maliciosos
- Caso 3: Encuentra el ataque conociendo un host sospechoso
- Caso 4: Identifica a los dominios potencialmente maliciosos (no más de 5 falsos positivos por día)

Para ver los datos y logs compartidos por el laboratorio de Los Álamos para este desafío se puede visitar el siguiente servidor FTP: ftp://ftp.lanl.gov/public/pflarr/

4.3. REQUERIMIENTOS DE PROCESADO

Hay unos cuantos aspectos muy importantes que se deben en cuenta para luego poder decidirnos por cuáles son los requerimientos de procesado que deberá tener la plataforma de cálculo y deducción lógica de este proyecto:

- Se necesita una gran cantidad de información de logs DNS previos a la infección de nuestro entorno para poder "entrenar" el sistema.

- La captura de datos y análisis será puntual e incremental, con lo que cada vez será mayor la cantidad de variables y texto en bruto que se obtendrán.

- Las APTs tienen un desarrollo a largo plazo y son "persistentes" en el tiempo, por lo que cabe esperar que la información DNS recogida a lo largo de ese periodo de tiempo será considerablemente grande.

- Las APTs se aprovechan enormemente del "ruido de entorno" generado por el resto de hosts de la red, por lo que la información aprovechable de hosts infectados puede llegar a ser incluso inferior a un 0,2% del total (2 hosts infectados de cada 1000). Una vez más esto supondrá que para poder sacar conclusiones satisfactorias será necesario analizar todo el volumen de tráfico diario.

- El tiempo de análisis debe ser inferior a un día, cada 24 horas debe empezar a procesarse el siguiente reporte recibido de la red.

- No sólo será necesario relacionar los hosts sospechosos de la red y los dominios visitados, sino también cuáles son otros hosts que han podido visitar dichos dominios y comunicaciones entre todos estos hosts con otros hosts no sospechosos (a priori) de la red.

Atendiendo a los anteriores aspectos se necesitan al menos los siguientes requerimientos:

- La plataforma de inferencia no puede estar basada en un único nodo
- El algoritmo de inferencia de la solución y del cálculo de probabilidad de que un host esté infectado o no debe estar distribuido.
- Se necesitará que sea una plataforma, que de forma simplificada, permita la compartición de ficheros de manera distribuida.
- El sistema debe además permitir un control muy introspectivo de las tareas que se están realizando, del rendimiento y del tiempo de ejecución (que debe siempre ser menor a 24h por ejecución).
- Será preferible (por la complicación que puede llevar el algoritmo) que sea un sistema estandarizado y con un paradigma de programación orientado a la distribución.
- Debido a la novedad de la problemática sería positivo que la plataforma sea una herramienta conocida y usada en la actualidad.
- Debido a que no es el objetivo único de este proyecto sería preferible que la plataforma ofrezca un fácil despliegue
- Los nodos que montemos deben estar en un mismo entorno de red y con enlace de alta velocidad de transmisión para minimizar los tiempos de cálculo.
- Sería positivo si los nodos que se van a montar pudieran estar basados en sistemas operativos de libre distribución, ya que el objetivo es que la solución de descubrimiento APT basado en DNS sea fácil de implementar sin incurrir en grandes gastos por parte de la empresa cliente.
- Debido a la limitada cantidad de recursos de las que se disponen en esta investigación sería también positivo si pudiera ser factible su implementación usando nodos de no mucha RAM.

4.4. PLATAFORMA HADOOP CON NODOS CENTOS

Por todo lo citado en el anterior apartado "Requerimientos de procesado" se ha elegido que la plataforma que se va a usar se base en **Hadoop** montado a partir de un servidor **Cloudera Manager** sobre sistemas **"CentOS minimal"**:

- Hadoop es un framework de licencia libre basado en tecnología Apache que sirve como soporte de aplicaciones distribuidas. Esto permite trabajar con enormes cantidades de información (BigData) de incluso petabytes. En su desarrollo

participan múltiples de colaboradores a nivel mundial y se usa Java como principal lenguaje de programación.

Este framework será muy útil debido a que la cantidad de datos que se van a procesar. A pesar de no ser de petabytes, sí que se va a necesitar tratar la información de forma distribuida para agilizar la inferencia. Más información en (Apache, s.f.) [11]

- CentOS es un tipo de sistema operativo derivado a nivel binario de la distribución Red Hat Enterprise Linux (RHEL). Al igual que Red Hat se compone de software libre y código abierto, pero a diferencia de éste no se presenta en forma de producto comercial con una suscripción, asistencia técnica y mantenimiento. Como el resto de sistemas Linux se caracteriza por el uso prioritario de la interfaz de comandos en lugar de la gráfica. De hecho, en la versión a utilizar ("minimal") no hay interfaz gráfica y esto permitirá que el consumo de RAM del SO sea menor y poder dedicar mayor parte al cálculo y la inferencia en el cluster. Más información en (CentOS, s.f.)[12]

- Cloudera Express es una solución de la empresa Cloudera (con siglas CDH, Cloudera Distribution Including Apache Hadoop) cuyo objetivo es facilitar el despliegue de la tecnología Hadoop en entornos de tipo empresarial. Esta solución consiste en una distribución de CDH a partir de un servidor Cloudera Manager que envía, organiza y orquesta la funcionalidad (roles), paquetes, repositorios y actualizaciones necesarias en el resto de nodos para que el cluster Hadoop pueda funcionar de forma satisfactoria. Más información en (Cloudera, s.f.)[13]

4.5. REQUISITOS DEL ALGORITMO DE INFERENCIA

El algoritmo que sea escogido debe tener en cuenta el cumplir ciertas características:

- Debe ser un algoritmo distribuible, ya que la plataforma es distribuida

- Debe estar basado en "estados" momentáneos de los nodos, ya que en los logs se registrará información por periodos (1 día, 1 semana, etc)

- Es necesario que la inferencia de la plataforma se pueda basar en "deducciones" o "creencias" sobre el estado (infectado/no infectado) de los nodos.

- Debe permitir que muchas relaciones entren en juego entre los diferentes nodos.

- Es necesario que el sistema completo sea convergente hacia una solución en la que la variación de los datos sea inferior a un rango previamente determinado, momento en el que consideraremos que el sistema es estable.

- Debe contemplar el establecimiento de pesos a los diferentes nodos y relaciones para categorizar qué nodos tienen una mayor peligrosidad y cuáles no.

4.6. ALGORITMOS CONTEMPLADOS

- **Fuzzy logic algorithms / algoritmos de lógica difusa**

En estos algoritmos las decisiones se basan en la decisión entre dos valores (en el caso aquí tratado podría ser infectado-no infectado) pero parcialmente referidos entre sí (en ocasiones puede ser subjetivo si un host está infectado o no). Un ejemplo muy utilizado en este tipo de algoritmos es el que determina mentalmente en el cerebro si una persona es alta o no. Normalmente se concibe que una persona de 1,90m es alta y que una persona de 1,50m es baja, porque el "conjunto" conocido de individuos ha configurado esta percepción en el cerebro a base de repetición, pero… ¿Se puede considerar que es alta una persona mide 1,72m? Se podría decir que "no es muy alta" o que "no es muy baja", en este tipo de asertos es en los que se basa la lógica difusa. Ejemplo de (Thórisson, s.f.)[14]:

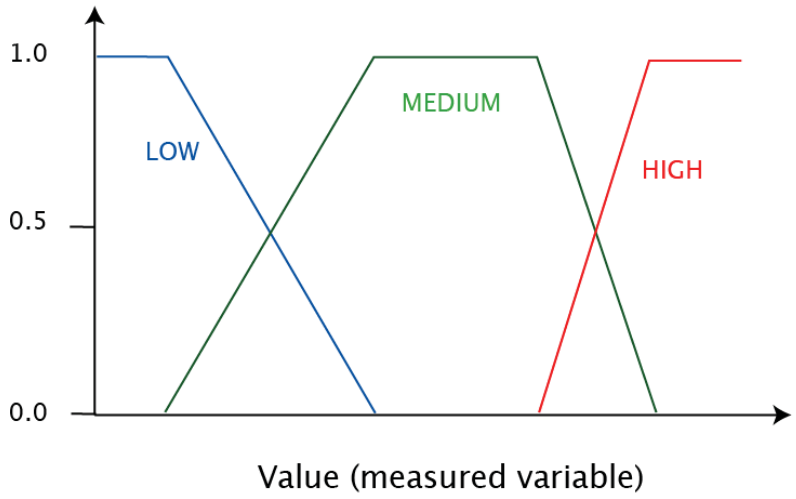

Figura 19: Ejemplo de valores contemplados por la lógica difusa

El mayor problema de este tipo de lógica es que, ajustándose a la vida real ofrece una gran cantidad de rangos posibles y realiza fluctuaciones de decisión en un margen de subjetividad, pero no nos permite tomar sólidas inferencias en base a pequeños periodos de recolección de información sin dar falso positivo. El algoritmo escogido debe garantizar que no habrá falsos positivos durante la recogida de datos y no puede tomar decisiones difusas, sino realizar sospechas y notificar cuando considere que están lo suficientemente contrastadas. Si bien a largo plazo y con un conjunto grande de muestras sería idóneo, el sistema necesita que el algoritmo infiera de manera convergente.

- **Multi-Entity Bayesian Network/PR-OWL**

Es un sistema mixto capaz de establecer inferencias sólidas en base a una ontología o base de conocimiento que bien puede contener elementos "difusos" o elementos ontológicos sólidos y conocidos (asertos).

La inferencia se realiza a partir de la base de conocimiento (mezcla de difusa y cierta) y la información recibida. Este sistema sería perfecto para la realización de una inferencia completa en base a suposiciones previas (lógica difusa), información segura (entidades completas) y la nueva información captada tras la captura de los nuevos logs.

Aunque no existe en la actualidad un mecanismo de inferencia capaz de tomar decisiones basándose en todos estos factores se podría llegar a implementar utilizando una arquitectura lógica parecida a la que aquí se plantea en el ejemplo extraído de (Cantarero Dávila & Ramírez Fernández, 2013)[15]:

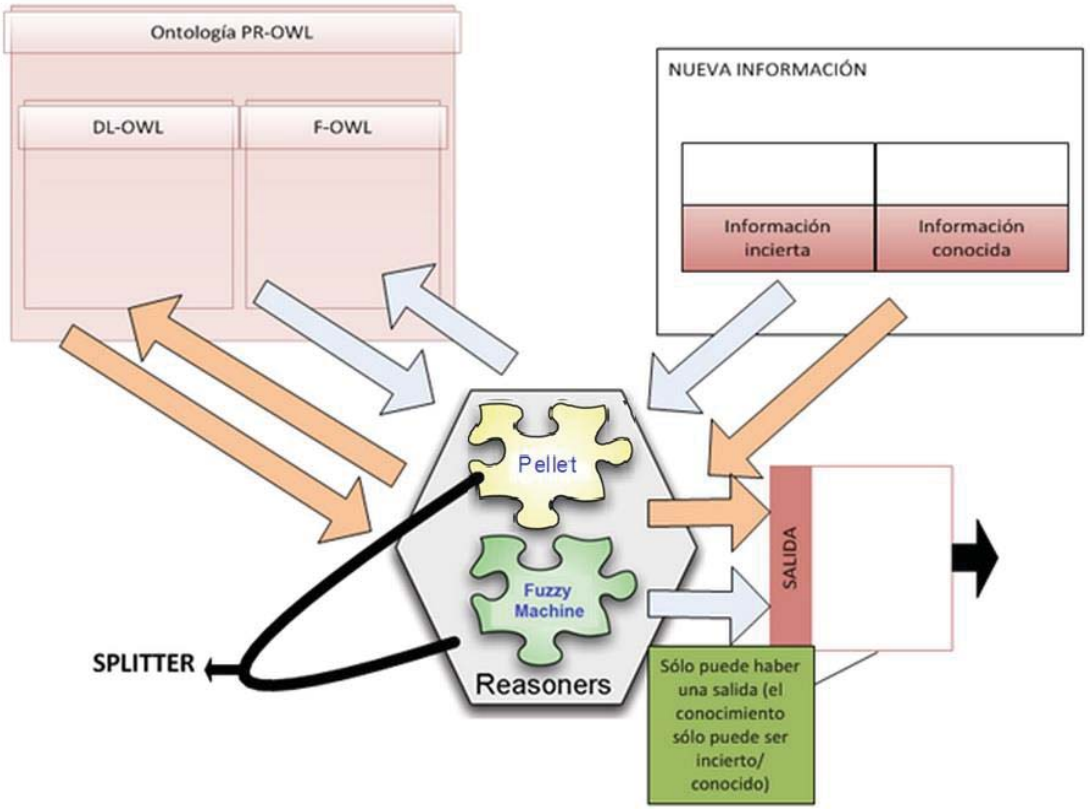

Figura 20: Arquitectura de inferencia propuesta para MEBN y PR-OWL

En cambio, debido a la complejidad de la arquitectura planteada y a que el mero desarrollo de una arquitectura de inferencia basada en MEBN podría suponer una tesis por sí misma se ha preferido descartar esta opción para la elaboración de este proyecto.

- **Belief Propagation**

Es un algoritmo de inferencia basado en el cálculo de las probabilidades marginales de cada estado de un nodo en relación con todos los demás que le rodean. El cálculo será más complejo cuantas más relaciones haya, pero no necesariamente cuanto más nodos haya en el conjunto de estudio.

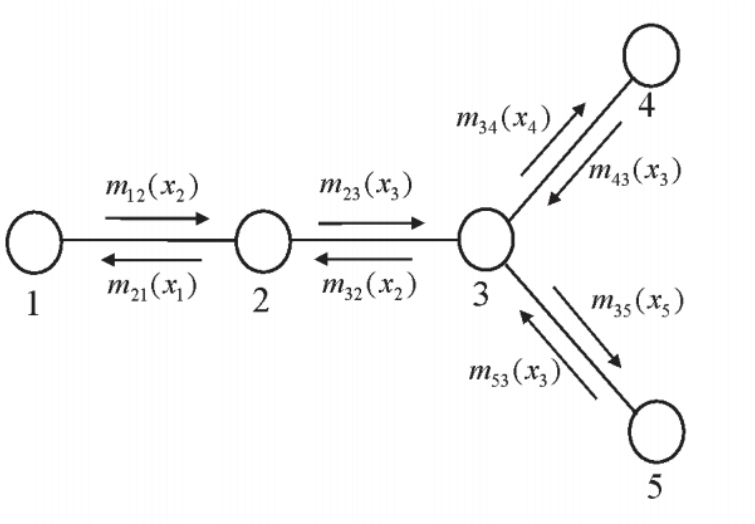

Figura 21: Ejemplo de mensajes intercambiados en un modelo Belief Propagation

$m_{ij}(x_j)$ es un mensaje desde un nodo i a un nodo j que nos revela en qué estado debería estar el nodo j. Esto puede ser simplificado a que, imaginando que cada uno de los hosts de nuestra red de estudio fuera uno de esos nodos (considerando a las URLs visitadas también como nodos), el estudio se basará en esos mensajes o relaciones entre nodos.

Los nodos del flujo se intercambian mensajes en iteraciones continuas. Nada impide establecer un flujo que contenga lazos, a no ser que la especificación del algoritmo así lo indique. La actualización de estados en base a la llegada progresiva de mensajes hará que el sistema fluctúe cada vez menos por aproximación al estado de estabilidad correspondiente más próximo. Para ello se deberán restringir los conjuntos de interacción y atender a los lazos que se han creado, ya que de manera contraria el sistema podría no converger.

Para poder usar este algoritmo en Hadoop se va a crear una versión propia del mismo de forma reducida al que se va a apodar “Light Belief Propagation” (LBP). De esta manera se compaginan la filosofía de los mensajes entre nodos y la relación convergente entre ellos. Se escogerán aquellas características que nos faciliten su implementación mediante un sistema distribuido.

4.7. MODELO MAPREDUCE

El algoritmo antes mencionado presenta la cualidad de ser fácilmente asimilable a la filosofía de Hadoop, MapReduce. MapReduce es un modelo de programación para la computación paralela de enormes cantidades de información (en este caso logs), también llamado BigData. Los modelos MapReduce dividen la información en datos estructurados en duplas del tipo clave-valor y les aplican dos funciones (map y reduce).

La estructura se divide en nodos master y nodos trabajadores. En la función Map() el nodo master coge la entrada de pares clave-valor y divide la tarea en problemas de menor envergadura que reparte a los nodos trabajadores (los que a su vez podrían volver a fraccionar el problema). Cada uno de los nodos distribuidos trabajadores calculará su sub-tarea por separado y devolverá una respuesta (otro ítem) al nodo maestro. La función Reduce() toma la lista de ítems clave-valor respuesta de los nodos trabajadores en una lista de valores final o en un único resultado (map&reduce).

Hay otras funciones intermedias que se realizan durante el proceso MapReduce, en concreto las de "Splitting" o separación de ítems antes de la función "Map" y del "Shuffling" o reorganización del contenido antes de la función "Reduce". Se puede observar un ejemplo extraído del artículo (Zhang, 2013)[16]:

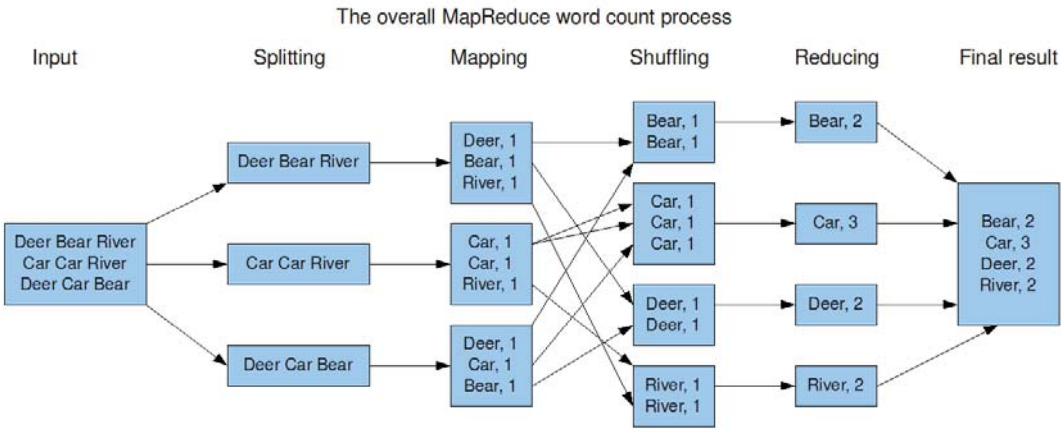

Figura 22: Ejemplo de proceso para contar palabras usando MapReduce

4.8. REDUCCIÓN AL MODELO CLAVE-VALOR

Tal y cómo se acaba de ver es necesario que la problemática sea reducida a un modelo en el que todo sea representable por vía de objetos compuestos por la dupla clave-valor. Se hacen los siguientes asertos:

- Se deberá obtener el número de visitas a cada URL (suma total)

- Se deberá extraer la lista de hosts que han visitado cada URL. Con esto y lo anterior se filtrará la lista considerablemente (una página poco visitada y sólo visitada por un host levanta sospechas y luego podremos comprobar si se realizan consultas periódicas).
- Se relacionara a cada host con sus horas de actividad según patrones(para detectar procesos repetitivos)
- Se deberán relacionar los hosts investigados con las franjas de horario laboral
- Se deberá relacionar qué hosts han contactado con otro host concreto (expansión lateral de la APT)
- Se deberá extraer que hosts han tenido más visitas de otros hosts

Una vez obtenidas todas estas relaciones se deberá asociar una puntuación de peso condicional encadenada y calcular la "creencia" de infección de cada nodo.

Tal y cómo se puede deducir de los asertos anteriores no se deduce un único tipo de objetos clave valor, sino todo un muestrario complejo:

Tabla 2: División en clave-valor de los aspectos de estudio de las APTs

Objetivo de estudio	**Comportamiento del APT**	**Clave MR**	**Valor MR**
Número de visitas a cada URL	• Callback	URL	Nº Visitas
Lista de hosts que visitan cada URL	• Callback	URL	Lista de hosts
Detectar patrones en actividades que sean susceptibles de ser ejecutadas por un proceso no-humano	• Escalado de privilegios • Callback • Mov. lateral	Patrón	Nº de repeticiones
Actividades fuera de horario laboral que puedan levantar sospechas sobre actividad no-humana y procesos automáticos	• Escalado de privilegios • Callback • Mov. lateral	Patrón	Nº de repeticiones
Hosts que intentan contactar con muchos otros	• Mov. lateral	Host	Lista de hosts

Hosts que actúan de callback y concentran información recabada de la red y robada del entorno infectado	• Mov. lateral	Host	Nº de visitas de otros hosts

Ante esta perspectiva de un algoritmo Hadoop con múltiples claves-valores se deduce que el algoritmo completo del proyecto deberá estar compuesto de varios "sub-algoritmos" para cada una de estas duplas.

Para cada una de las iteraciones se tendrían que ejecutar todos estos algoritmos en una sucesión para al final poder otorgar un peso a los resultados de los diferentes procesos MapReduce y obtener en cada uno de los nodos un valor de "creencia" de infección en base a las experiencias no nodos vecinos (ya sean esos nodos horas, webs, otros hosts…)

4.9. LIGHT BELIEF PROPAGATION

El objetivo del algoritmo no es otro que el de descubrir cuáles son los hosts infectados de una red. Para ello se van a hacer las siguientes consideraciones al respecto:

- Cada host de la red será un nodo en el modelo probabilístico
- Cada URL visitada diferente será un nodo en el modelo probabilístico
- Se contrastará (para simplificar el algoritmo) la lista de URLs visitadas con una base de datos de páginas infectadas para comprobar si la APT usa repositorios maliciosos conocidos
- Se contrastarán las URLs visitadas en los últimos días con todas las URLs visitadas desde que está en marcha el sistema
- Se comprobarán patrones de fecha y reiteración (nodos que intercambian comunicación a horas coincidentes o con cierta periodicidad)
- Se comprobará que las horas de tráfico de hosts de usuario no son fuera del horario laboral
- Por cada vez que se ejecute el algoritmo se deberán entonces ejecutar los siguientes cálculos MapReduce:
    - URL-Nº visitas
    - URL-Lista de hosts
    - Patrón horario-Nº de repeticiones
    - Patrón de franja laboral-Nº de repeticiones
    - Host – Lista de hosts

  - Host – Nº de visitas recibidas

- Cada vez que se ejecute uno de estos algoritmos se distribuirá el trabajo según el modelo MapReduce y la lectura de datos y carga en memoria se tendrá que hacer tantas veces como sub-algoritmos se ejecuten (en este caso 6).
- Aquellas URLs visitadas infectadas se marcarán con un índice de peso de 1,1
- Los hosts que visitaron las URLs infectadas serán marcados con un peso de 0,9
- Los hosts que presentan comportamientos anómalos periódicos se marcarán con un peso de 0,8
- Los hosts que presentan un comportamiento fuera del horario laboral serán marcados con un peso de 0,7
- Los hosts que establecen comunicación con hosts con un peso superior o igual a 0,7 incrementarán su peligrosidad atendiendo a la fórmula:

$$PesoActualizado=PesoViejo+PesoNodoPeligroso*0,1$$

- El cálculo del algoritmo se ejecutará día a día y la fórmula será actualizada siguiendo los pesos atendiendo a la expresión anterior
- Si el host lleva dos días seguidos sin comportamientos anómalos:

$$PesoActualizado=PesoViejo+0,1$$

- Se detendrá el algoritmo para un nodo concreto (y se pasará a ser infectado o no infectado) cuando varíe su peso en una diferencia total inferior a un 1% del valor durante 5 iteraciones seguidas.
- De aquí se obtiene el siguiente pseudocódigo a alto nivel:

```
Inicio del programa desde nodo Hadoop(URIlogsDNS,
patron_horario, horario_laboral,
tareas_programadas_fuera_horario_laboral)

Leer datos de logs de DNS del Path pasado por parámetro
Guardar como variable los parámetros de patron_horario
Guardar como variable los parámetros de horario_laboral
Extraer URLs visitadas como una lista
Comprobar lista de URLs visitadas con lista de webs
sospechosas en internet
Ejecutar Job MapReduce(URL-n_visitas) //Tenemos el UML
Ejecutar Job MapReduce(URL-lista_hosts) //Tenemos el UML
```

```
Ejecutar Job MapReduce(patron_horario-n_repeticiones)
Ejecutar Job MapReduce(host-lista_hosts)
Ejecutar Job MapReduce(host-n_visitas)

Calcular_pesos:
     Si el host presenta comportamiento periódico anómalo
               Si tiene peso
                    Se multiplica su peso por 0,8
               Si no
                    Se asigna su peso a 0,8
               Finsi
               Se considera comportamiento sospechoso
     Finsi


     Si el host presenta comportamiento injustificado
fuera de horario laboral:
               Si tiene peso
                    Se multiplica su peso por 0,7
               Si no
                    Se asigna su peso a 0,7
               Finsi
               Se considera comportamiento sospechoso
     Finsi
     Si el host establece comunicacion con un host con
peso inferior a 0,7:

     PesoActualizado=PesoViejo+PesoNodoPeligroso*0,1
               Se considera comportamiento sospechoso
     Finsi
     Si el host lleva 2 dias sin comportamiento sospechoso
               PesoActualizado=PesoViejo+0,1
     Finsi

     Si
          (peso_hace5_iteraciones-
          PesoActualizado<PesoActualizado/100)
          y (peso_hace4_iteraciones-
          PesoActualizado<PesoActualizado/100)
          y (peso_hace3_iteraciones-
          PesoActualizado<PesoActualizado/100)
```

```
                y (peso_hace2_iteraciones-
                PesoActualizado<PesoActualizado/100)
                y (peso_hace1_iteraciones-
                PesoActualizado<PesoActualizado/100)

        Entonces:

        Peso_final=PesoActualizado
        Finsi
Fin_Calcular_pesos

Devolver la lista de pesos //Los nodos con peso inferior a
//0,7 se consideran infectados
```

4.10. TOPOLOGÍA DE RED E INFRAESTRUCTURA

Para montar esta plataforma se va a usar un entorno virtualizado bajo la plataforma VMWare Workstation. La maqueta estará compuesta por 4 nodos virtualizados. Se va a necesitar que los servidores virtuales tengan características físicas similares porque serán nodos de un mismo cluster en el cuál es importante que se agilice el proceso por recargo equitativo de carga.

Los recursos que se usarán son:

Sistema anfitrión:

Tabla 3: Resumen de recursos hardware del sistema anfitrión

Campo	Valor
Tipo de equipo	Equipo basado en x64 de ACPI
Sistema operativo	Microsoft Windows 8.1
Placa base	
Tipo de CPU	QuadCore Intel Core i7-3770, 3900 MHz (39 x 100)
Nombre de la placa base	Pegatron 2AD5
Chipset de la placa base	Intel Panther Point Z75, Intel Ivy Bridge

Memoria del sistema	16323 MB (DDR3-1600 DDR3 SDRAM)
DIMM1: Kingston HP698650-154-HYAG	4 GB DDR3-1600 DDR3 SDRAM (11-11-11-28 @ 800 MHz) (10-10-10-27 @ 761 MHz) (9-9-9-24 @ 685 MHz) (8-8-8-22 @ 609 MHz) (7-7-7-19 @ 533 MHz) (6-6-6-16 @ 457 MHz)
DIMM2: Kingston HP698650-154-HYAG	4 GB DDR3-1600 DDR3 SDRAM (11-11-11-28 @ 800 MHz) (10-10-10-27 @ 761 MHz) (9-9-9-24 @ 685 MHz) (8-8-8-22 @ 609 MHz) (7-7-7-19 @ 533 MHz) (6-6-6-16 @ 457 MHz)
DIMM3: Kingston HP698650-154-HYAG	4 GB DDR3-1600 DDR3 SDRAM (11-11-11-28 @ 800 MHz) (10-10-10-27 @ 761 MHz) (9-9-9-24 @ 685 MHz) (8-8-8-22 @ 609 MHz) (7-7-7-19 @ 533 MHz) (6-6-6-16 @ 457 MHz)
DIMM4: Kingston HP698650-154-HYAG	4 GB DDR3-1600 DDR3 SDRAM (11-11-11-28 @ 800 MHz) (10-10-10-27 @ 761 MHz) (9-9-9-24 @ 685 MHz) (8-8-8-22 @ 609 MHz) (7-7-7-19 @ 533 MHz) (6-6-6-16 @ 457 MHz)
Tipo de BIOS	AMI (12/22/11)
Particiones	
C: (NTFS)	916.4 GB
D: (NTFS)	13635 MB
J: (NTFS)	465.8 GB
Tamaño total	1395.4 GB
Red	
Dirección IP primaria	192.168.56.1
Dirección MAC primaria	0C-54-A5-05-31-0E

Adaptador de red	Adaptador virtual directo Wi-Fi de Microsoft
Adaptador de red	Controladora Gigabit Ethernet Qualcomm Atheros AR8161 PCI-E (NDIS 6.30) (192.168.1.36)
Adaptador de red	Ralink RT5390R 802.11bgn Wi-Fi Adapter
Adaptador de red	VirtualBox Host-Only Ethernet Adapter (192.168.56.1)
Adaptador de red	VMware Virtual Ethernet Adapter for VMnet1 (192.168.74.1)
Adaptador de red	VMware Virtual Ethernet Adapter for VMnet8 (192.168.58.1)

Se necesitaran 6 IPs virtuales (una para el anfitrión, otra para el vSwitch y 4 para los hosts virtualizados). Se tendrá también un servidor de nombres en el que se han de introducir entradas DNS:

Tabla 4: Reparto de nombres-IPs en el cluster Hadoop

Host	Nombre DNS	IP
Anfitrión/Anfitrión	-	192.168.58.1 /192.168.58.2
Host 1	ns1.apt.edu	192.168.58.129
Host 2	cmanager.apt.edu	192.168.58.130
Host 3	clouderaclient1.apt.edu	192.169.58.131
Host 4	clouderaclient2.apt.edu	192.169.58.132

Para hacer una aproximación más cercana a la realidad se va a instalar en el cluster todas las herramientas que tendría una plataforma Hadoop completa: HDFS, MapReduce, ZooKeeper, Hive, Hue, Oozie y Cloudera Management Services.

Todas estas tareas estarían muy repartidas de ser este un cluster con un número importante de nodos, pero en este caso se van a concentrar más de 3 tareas en cada host, por lo que será necesario en cada host un poco más de RAM y no podremos montar nodos que tengan una reducida capacidad de memoria.

Inicialmente se va a hacer un breve resumen de cada uno de los servicios antes nombrados para tener una perspectiva al menos superficial de para qué sirve cada uno de ellos:

- **HDFS:** Es un sistema de archivos distribuido fabricado para correr en hardware de forma paralela y que permite la computación de una gran cantidad de datos a un muy bajo coste. Es muy tolerante a fallos y facilita la transacción muy rápida de información a pesar de que los datos transferidos sean de un tamaño muy grande. Los roles principales en HDFS son "NameNode" y DataNode", que son similares al conocido paradigma de maestro/esclavo. Los NameNodes manejan el espacio de nombres del sistema de ficheros y regulan el acceso a los ficheros a los clientes. Los DataNodes se encargan de leer y escribir en el sistema de ficheros y de crear bloques o trozos de información del tamaño indicado por el NameNode y proceder en cada situación a su borrado, replicado, traspaso a otro NameNode, etc. Un ejemplo de esta estructura sería el esquema extraído de (Hadoop, 2008)[17] en el que se observa cuál es la jerarquía y las operaciones que ejecuta cada nodo:

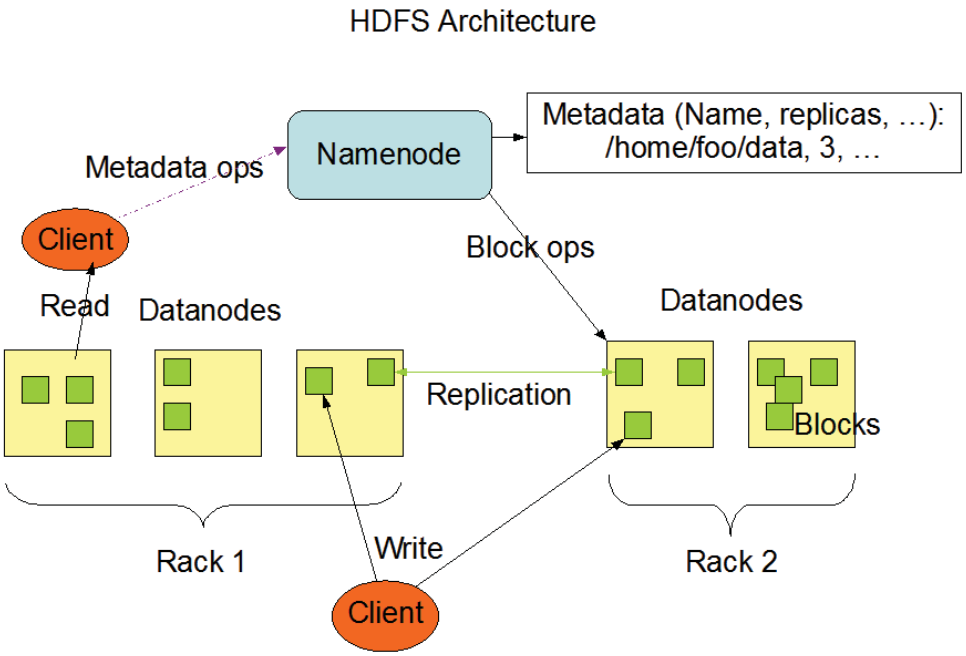

Figura 23: Esquema de funcionamiento del sistema HDFS

- **MapReduce:** Ya se ha comentado antes brevemente este paradigma de programación y repartición del trabajo, por lo que no se va a volver a insistir en este punto. Los dos roles principales son los de JobTracker (master) y TaskTracker. El master se encarga de programar las tareas básicas de cada TaskTracker, las monitoriza y vuelve a ejecutar aquellas que fallan. El esclavo (TaskTracker) tiene como única función la de ejecutar la tareas ordenadas por el master. Este esquema muestra cómo se distribuyen jerárquicamente las tareas en MapReduce:

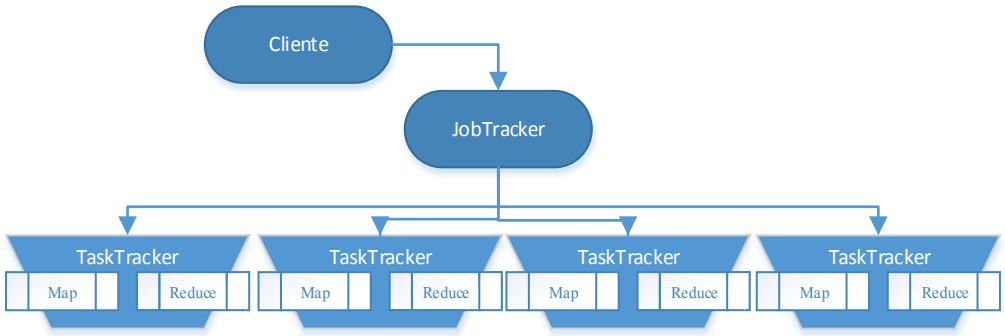

Figura 24: Esquema de funcionamiento de los roles de MapReduce

- **Hive:** Esta herramienta consiste en un una transformación del sistema Hadoop HDFS en un lenguaje basado en SQL que mantiene compatibilidad completa con el modelo Map/Reduce. Además acelera las consultas y provee un sistema de indexación basado en mapas de bits. Supone una manera fácil de manejar la base de datos HDFS con sentencias del tipo SQL sin abandonar el paradigma de la distribución Hadoop. Los roles diferentes de Hive son los de Gateway (ese nodo podrá transmitir la información SQL→HDFS), el Hive Metastore Server (almacena los metadatos del servicio en una base de datos relacional y permite el acceso a los clientes a la información a través de una API) y el WebHCat Server (provee la interfaz web de usuario). A pesar de que el sistema Hive genera un Job completo que remite al sistema Map/Reduce no se debe confundir con la tarea del NameNode. La tarea deberá seguir siendo dirigida por el nodo master de nuestro cluster Map/Reduce y procesada en bloques por los JobTracker. Imagen extraída de: (Shankar, 2011)[18]

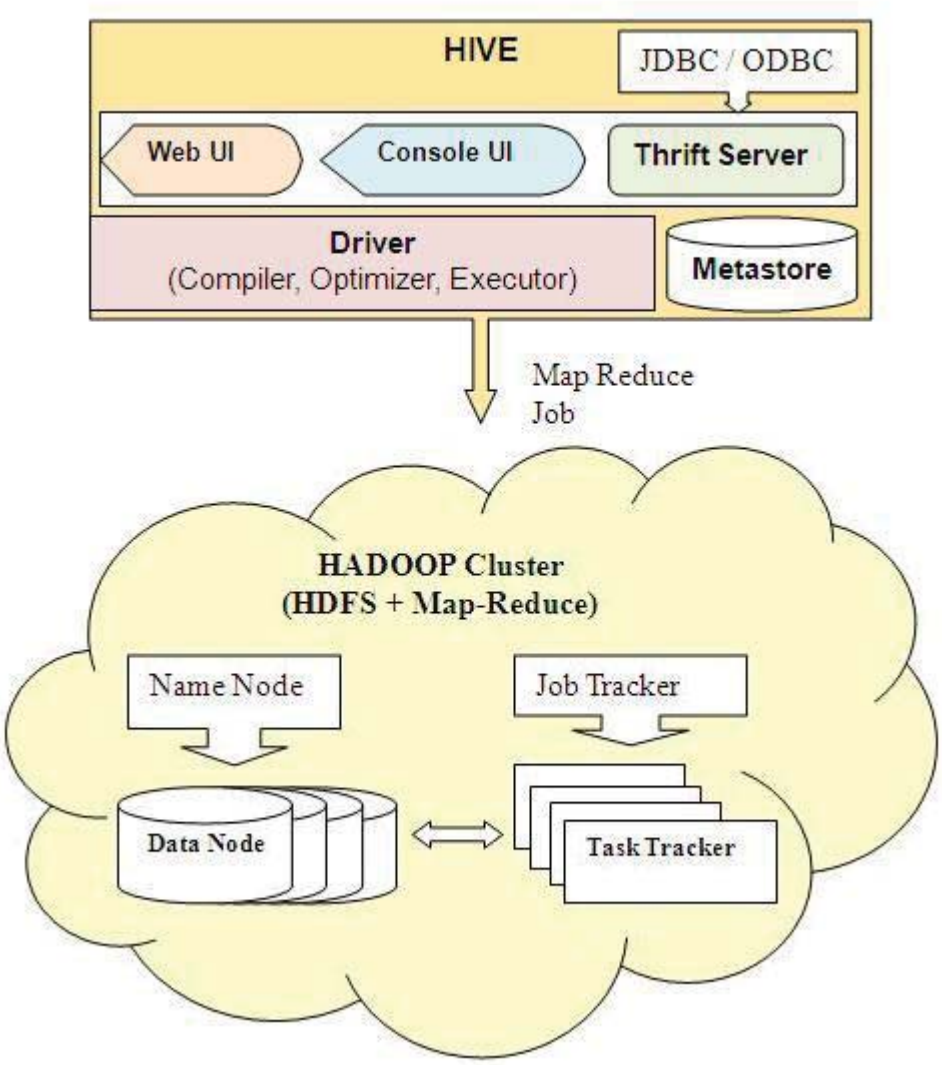

Figura 25: Esquema de funcionamiento de la herramienta Hive

- **ZooKeeper:** Es una herramienta que se encarga de ofrecer servicios de sincronización, nombrado de aplicaciones y mantenimiento de la configuración de las mismas que permiten al desarrollador abstraer la complejidad de todas las capas de la infraestructura distribuida montada. Sirve de orquestador y coordinador de la plataforma completa mediante una serie de primitivas que las aplicaciones distribuidas manejan y se comparten entre sí. Se puede obtener más información en (BDI Systems, s.f.)[19]

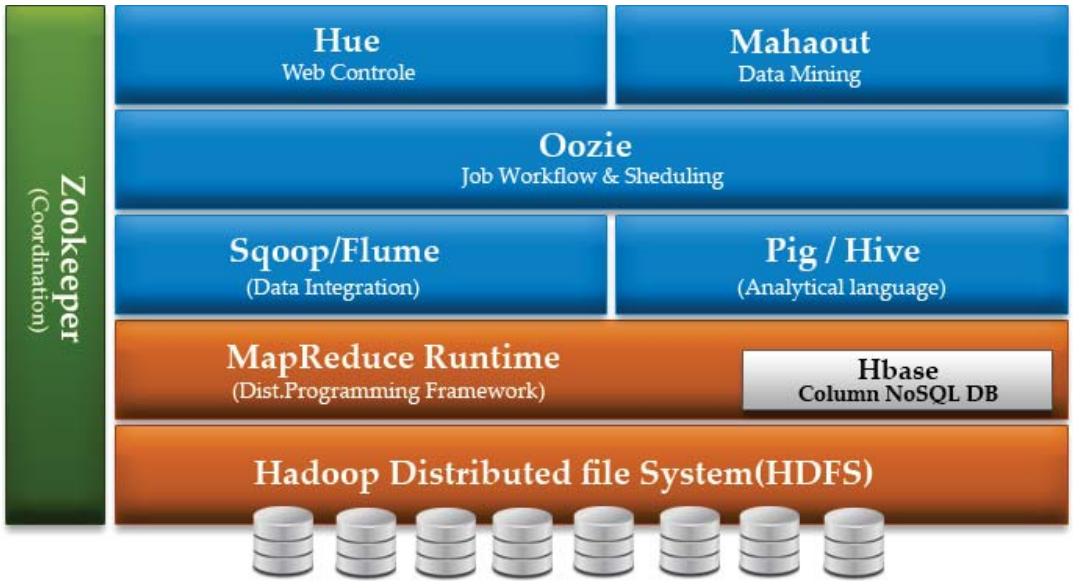

Figura 26: Esquema de arquitectura de plataforma Hadoop orquestada por Zookeeper

- **Oozie:** Es un programador de flujos de trabajo para gestionar las tareas de un cluster Hadoop. Se representa mediante gráficos no cíclicos en los que se encolan las diferentes tareas y se integra con el resto de la plataforma pudiendo soportar cualquier tipo de tarea (desde paradigma MapReduce en Java hasta a trabajar en Streaming). Es un sistema escalable, fiable y modular que ofrece gran cantidad de ventajas a los desarrolladores y usuarios de Hadoop, mejorando el rendimiento de una forma sencilla.
- **Hue:** Es una interfaz web para analizar datos de Hadoop y gestionar la plataforma, Jobs, datos y ficheros. Permite al usuario olvidarse de las interfaces de comandos y le otorga un navegador para observar el sistema HDFS

Después de ver los tipos de servicios, se necesitará hacer una distribución lógica de la RAM total del equipo para que cada máquina virtual pueda dar un rendimiento considerable y ninguna tenga mayor ratio de paginación que otra. Para conseguir esto se debe hacer una valoración de cuántos procesos va a llevar cada máquina y distinguir aquellos que serán cruciales para la asignación de tareas en el cluster y aquellos que tan sólo van a tener una labor de ejecutar procesos bajo demanda. Así es como van a ser repartidos los roles en esta arquitectura de cluster:

Tabla 5:Reparto de roles y RAM por nodos en el cluster Hadoop

Host	Nombre DNS	Tareas*	RAM
Host1	ns1.apt.edu	• **HDFS:** Secondary Namenode, DataNode • **MapReduce:** TaskTracker • **Hive:** Gateway, HiveServer2	2GB
Host2	cmanager.apt.edu	• **HDFS:** Namenode, HttpFS, DataNode • **MapReduce:** JobTracker, TaskTracker • **Hive:** Gateway, Hive Metastore Server, WebHCat Server • **ZooKeeper:** Server • **Hue:** Hue Server • **Oozie:** Oozie Server • **Cloudera Management Services:** Service Monitor, Activity Monitor, Host Monitor, Reports Manager, Event Server, Alert Publisher, Navigator Audit Server	3GB
Host3	clouderaclient1.apt.edu	• **HDFS:** DataNode • **MapReduce:** TaskTracker • **Hive:** Gateway	2GB
Host4	clouderaclient2.apt.edu	• **HDFS:** DataNode • **MapReduce:** TaskTracker **Hive:** Gateway	1,8GB

Nota: En un cluster Hadoop normal estas tareas estarían todas mucho más repartidas, pero al tener este cluster tan pocos hosts, el número de tareas por host se ve aumentado considerablemente.

Además de estos roles dentro del cluster, habrá dos máquinas (cmanager.apt.edu y ns1.apt.edu) que realizarán tareas extra:

Tabla 6: Tareas extra no relacionadas con el cluster Hadoop

Host	Nombre DNS	Tareas extra*
Host1	ns1.apt.edu	• Servidor DNS de todo el cluster
Host2	cmanager.apt.edu	• **Cloudera Manager:** Herramienta para el despliegue y la instalación del cluster de forma distribuida • Tendrá interfaz gráfica (si no la instalación se complicaría demasiado)

Cómo servidor DNS se ha usaremos el demonio de Linux named(bind9) y para instalar el servidor Cloudera Manager se hará desde el repositorio propio de cloudera. Se puede consultar en apartados posteriores el proceso de instalación de la plataforma.

Además, en el apéndice "Planos" se puede consultar entre los diferentes diagramas y esquemas una visión de "Despliegue" de la plataforma tanto en formato UML como especificando detalles de implementación virtual.

Se puede obtener una visión aproximada del coste que una arquitectura como esta podría suponer en el apartado "Presupuesto". Si bien, tal y cómo hemos comentado antes, esta plataforma no es de uso profesional y por tanto su coste es ridículo con una versión comercial, podremos hacer una aproximación del coste total que podría suponer su implementación en un entorno de producción.

5. PLANOS Y DIAGRAMAS

5.1 ESQUEMA DE LA RED VIRTUALIZADA PARA CLUSTER HADOOP

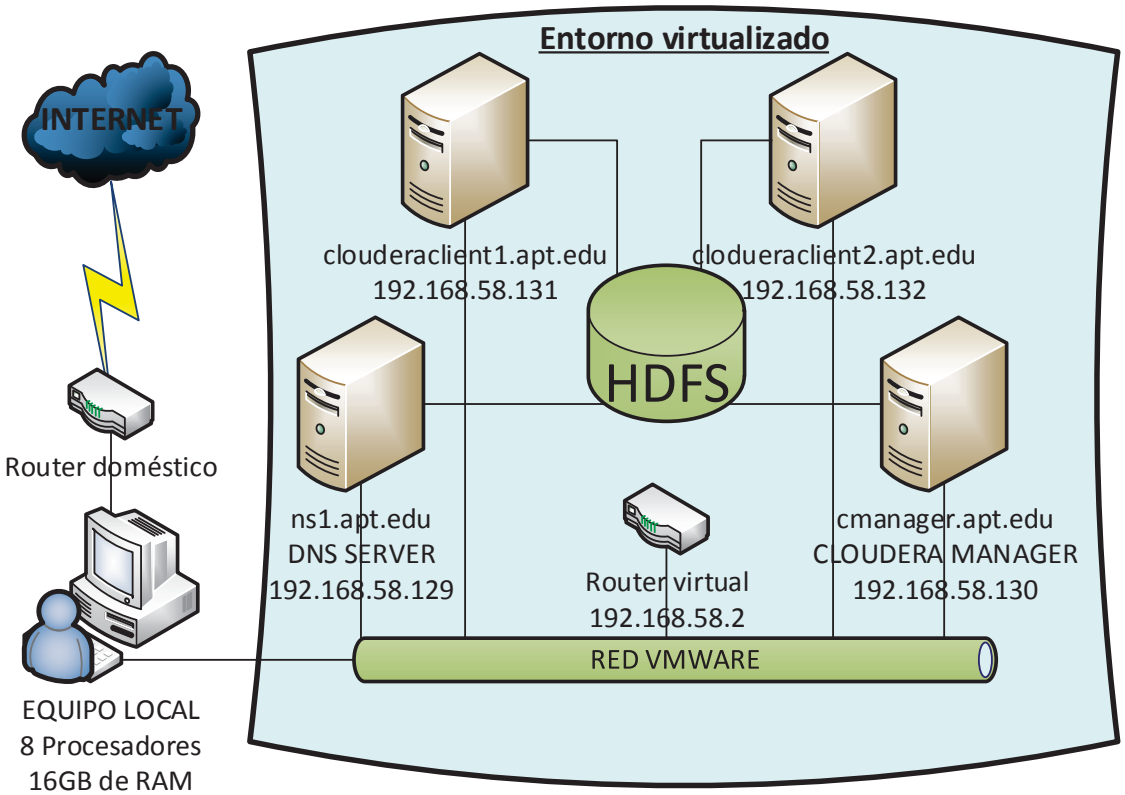

Figura 27: Esquema de elementos de red en nuestra plataforma Hadoop

En la figura superior se puede observar cuál es la configuración del equipo anfitrión para poder realizar la instalación de los hosts virtuales. Los elementos son los siguientes:

- Equipo local: Es el equipo físico anfitrión en el que se van a hospedar máquinas virtuales que simularán el entorno distribuido
- Rúter doméstico: Es un rúter físico ubicado en el domicilio. A diferencia del "rúter virtual" tiene conexión directa con Internet.
- Rúter virtual: Es un rúter simulado en el interior del entorno virtualizado para que sirva de "default Gateway" de los hosts virtuales que se encuentran dentro. Es el punto de comunicación entre el equipo físico y la red virtual de hosts.
- Cloudera Manager: Nodo encargado de distribuir la instalación de la plataforma Hadoop y de monitorizar al resto
- DNS Server: Host que almacenará las resoluciones DNS del resto de equipos virtuales para facilitar el direccionamiento. Puede además servir

de DNS corporativo en caso de que la infraestructura estuviera montada directamente en un cliente potencial si la herramienta fuera comercial.

- Red VMWare: Es la simulación de una red real que utiliza una interfaz específica entre varias disponibles (VMNet) para dar a los hosts virtuales conexión entre sí y proporcionar diferentes servicios (NAT, Bridging, isolated…)
- HDFS: Disco compartido entre los diferentes hosts que componen una infraestructura Hadoop. En caso de que fueran hosts físicos cada parte de HDFS perteneciente a cada máquina estaría distribuida físicamente en cada equipo. En este caso recae sobre los discos de cada uno de los hosts virtuales pero este a su vez vuelve a recaer sobre el disco físico del equipo local anfitrión.

5.2. DIAGRAMAS UML

Diagrama de secuencia

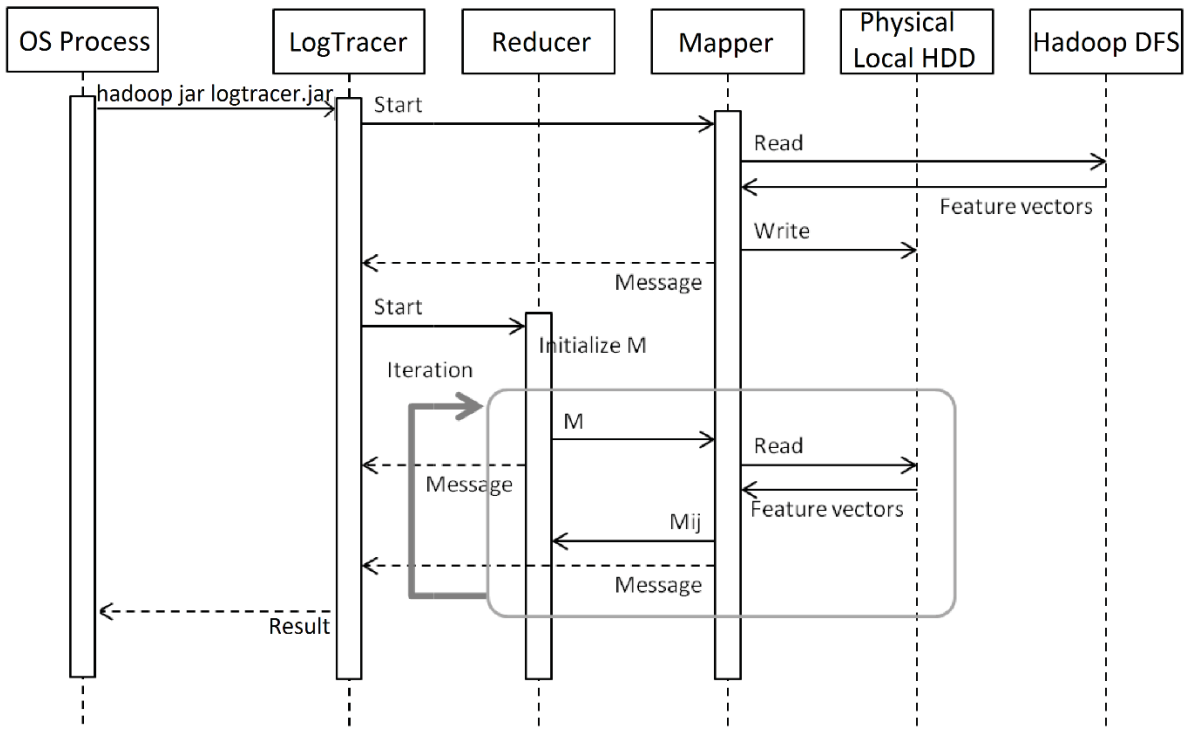

Figura 28: Diagrama de secuencia del sistema

La imagen superior muestra el diagrama de secuencia genérico de Hadoop particularizado para este caso. Tal y cómo se puede ver, primero se inicia el programa principal, luego se inicia la tarea de Map y luego la de Reduce de manera que entre ellas se intercambien las claves-valores.

Diagrama de casos de uso

Aunque no se ha implementado el algoritmo en una plataforma como si se tratara de un sistema preparado para el uso comercial aquí se explican cuáles serían las funcionalidades finales del sistema en caso de habilitar una API y de convertir todo el cluster en una plataforma "en la nube"

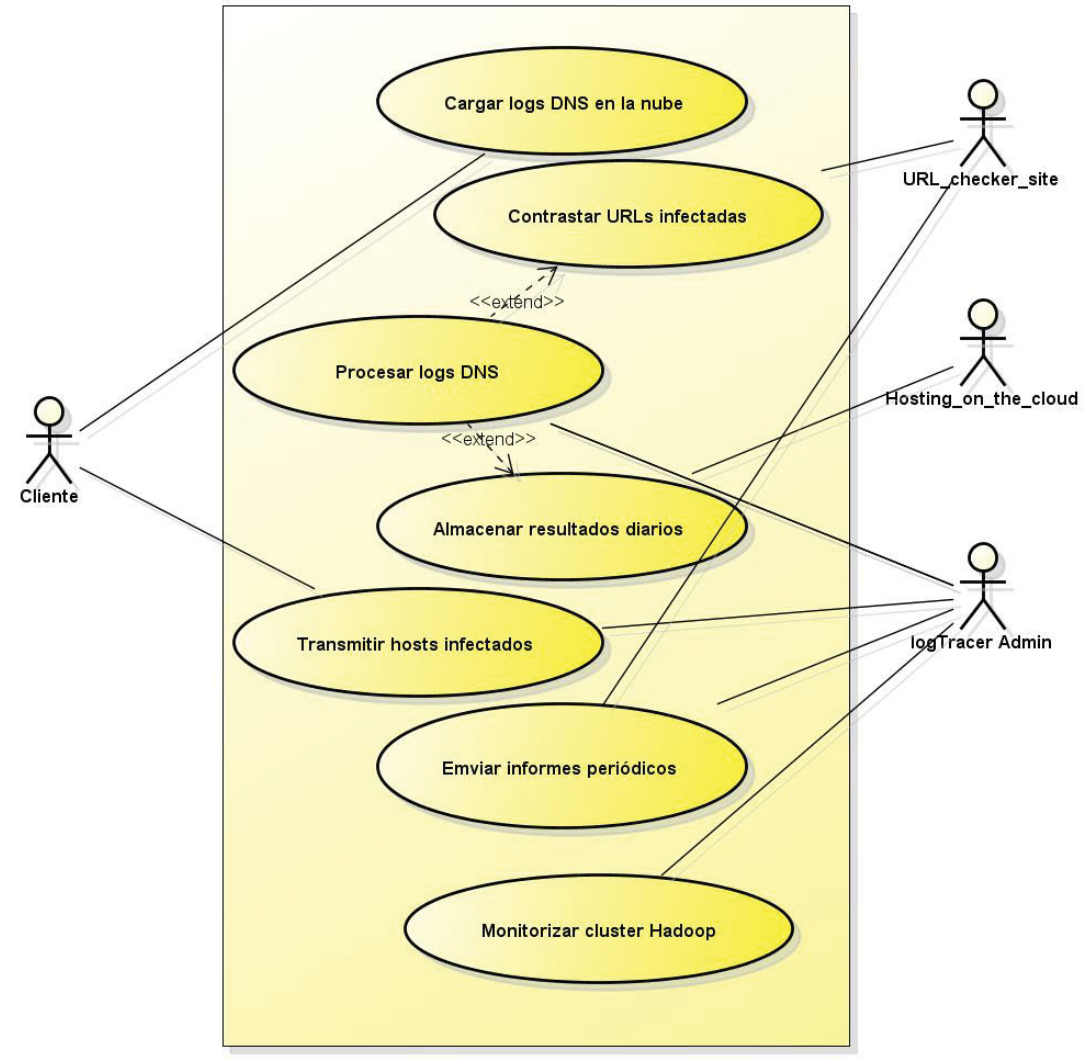

Figura 29: Diagrama de casos de uso del sistema logTracer

Diagrama de distribución

Se podría haber realizado un diagrama de distribución sencillo basado en la distribución de los servicios entre las 4 máquinas virtuales, pero en este caso se ha decidido hacer una abstracción completa del sistema y tener una visión completa desde el propio sistema anfitrión:

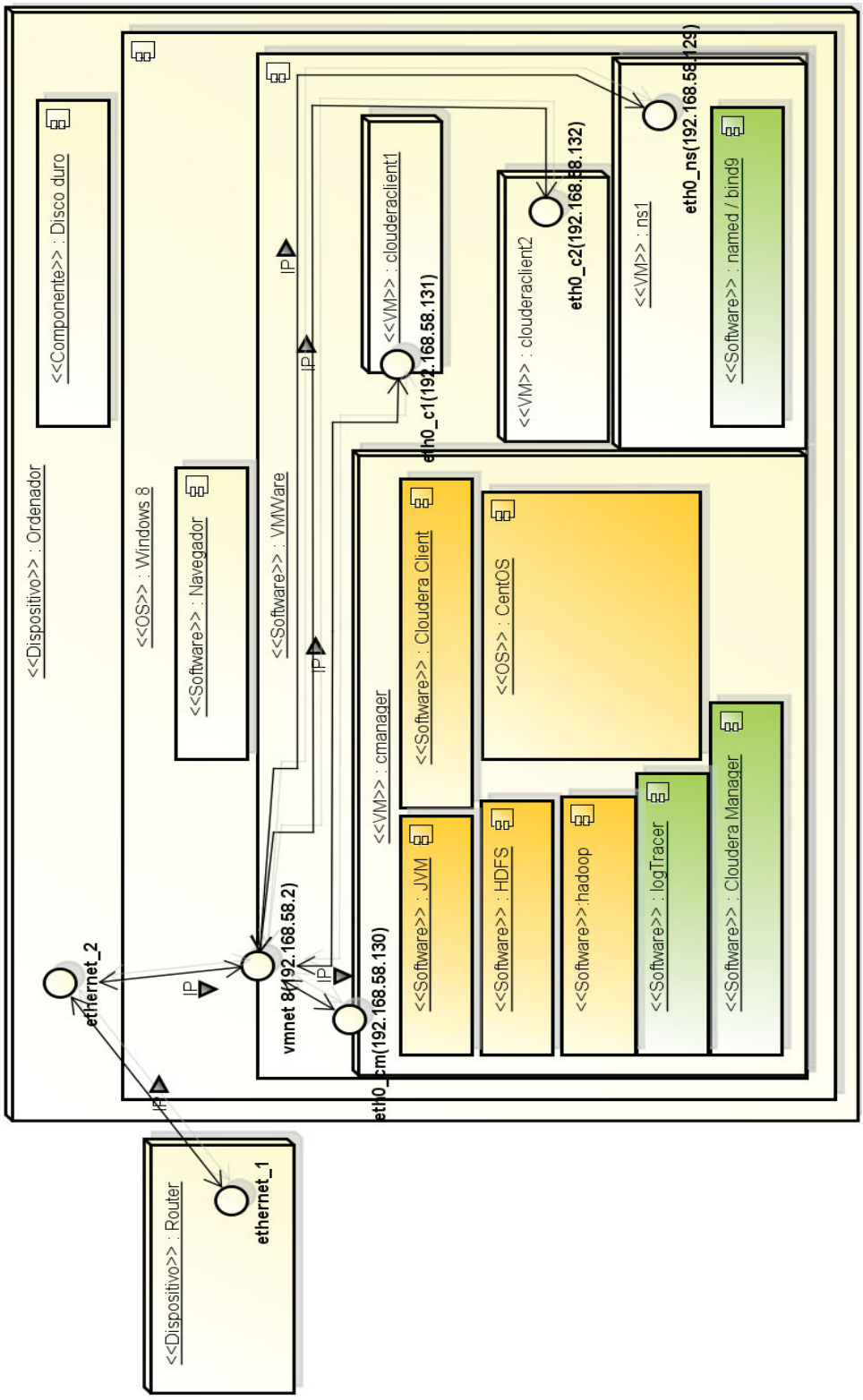

Figura 30: Diagrama de despliegue de la arquitectura propuesta

NOTA: Tal y cómo se observa en el gráfico anterior (diagrama de distribución de la plataforma) hay componentes de última instancia en los que se ha usado el color naranja para designar que son utilizados en el resto de máquinas virtuales (VM) y algunos en los que se ha usado el

color verde como indicativo de que ese componente es especial de esa máquina virtual en concreto.

Diagrama de componentes

Se ha tratado de limitar el diagrama de componentes al ámbito de cada máquina virtual ya que el diagrama de distribución ya debería dejar suficientemente claro cómo está repartido el sistema:

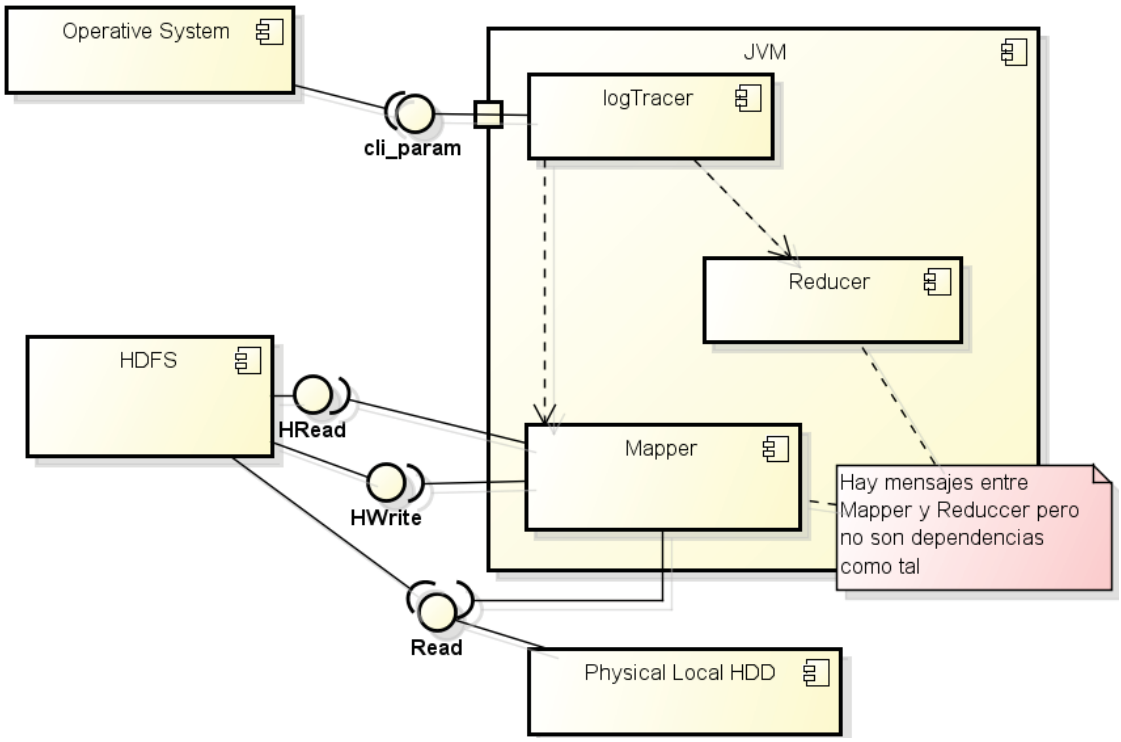

Figura 31: Diagrama de componentes local del sistema

Diagrama de clases

La imagen inferior muestra una parte del diagrama UML de clases que sería necesario para la elaboración de un programa que, basado en el paradigma MapReduce de Hadoop, sea capaz de calcular el objetivo perseguido.

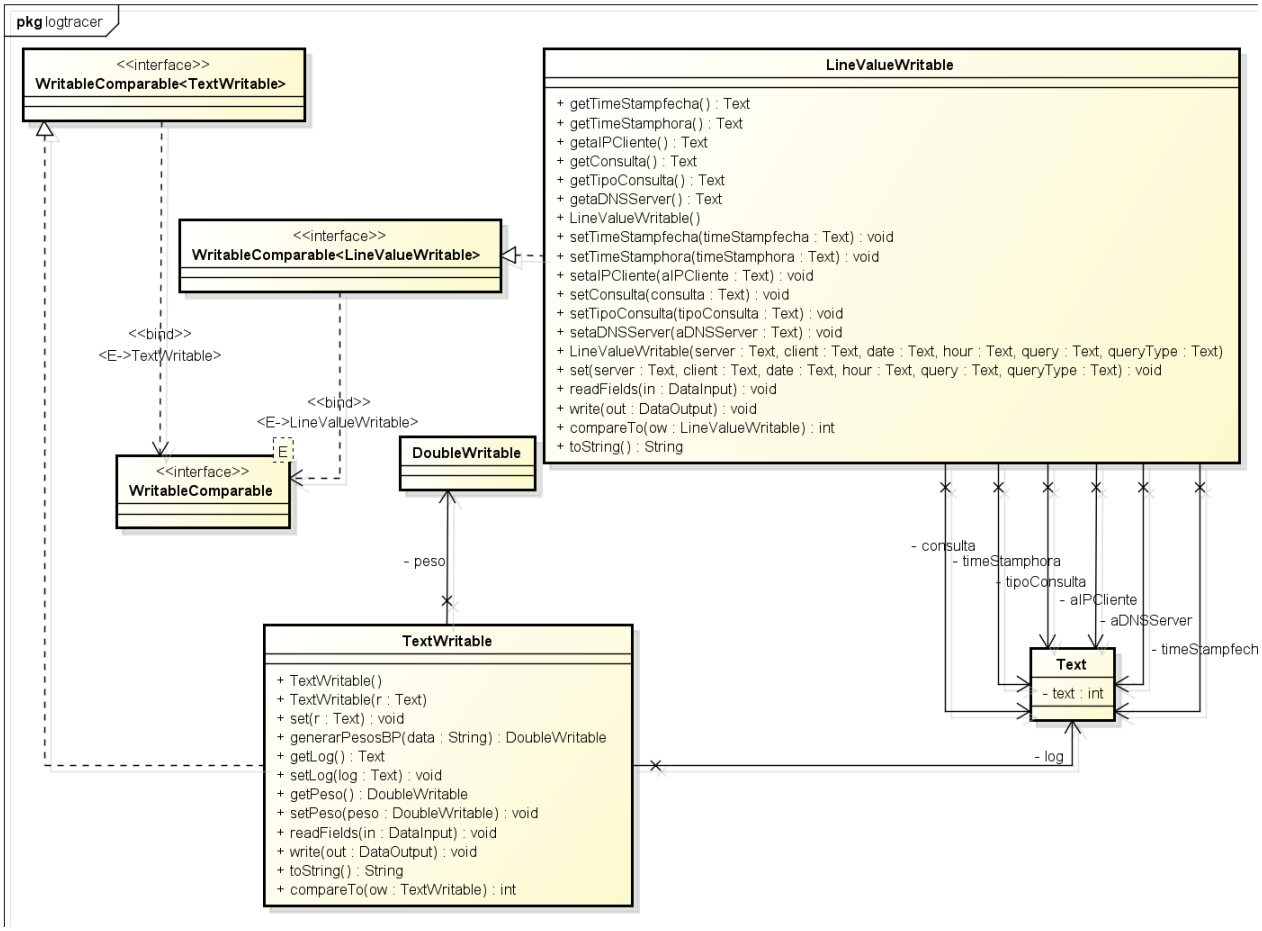

Figura 32: Diagrama de clases de los elementos"Writable" de la codificación

La mayoría de las clases aquí mostradas son mensajes y encapsulaciones usadas por el paradigma MapReduce para poder facilitar el cálculo de las etapas:

- WritableComparable: Sirve para poder compararse (típicamente vía Comparators) con otros WriteComparable. Todo objeto que vaya a ser usado como clave en un modelo Map-Reduce de implementar esta interfaz
- TextWritable: Cumple las funciones de tratamiento de texto en el paradigma MapReduce
- LineValueWritable: Cumple las funciones de tratamiento de una línea de texto en el paradigma MapReduce. Es de especial importancia ya que el tratamiento de la información será línea a línea (se sobreentiende que cada entrada de log estará en una línea) y porque tendremos que "parsear" cada uno de los elementos de la línea.
- Text: Pequeño elemento que puede representar una letra, una palabra, un trozo de línea o hasta un párrafo o escrito completo. Es utilizado para ser transferido como parámetro y en variables y manejar el texto de una forma más intuitiva (tiene métodos para transformar/parsear la variable en otros tipos).

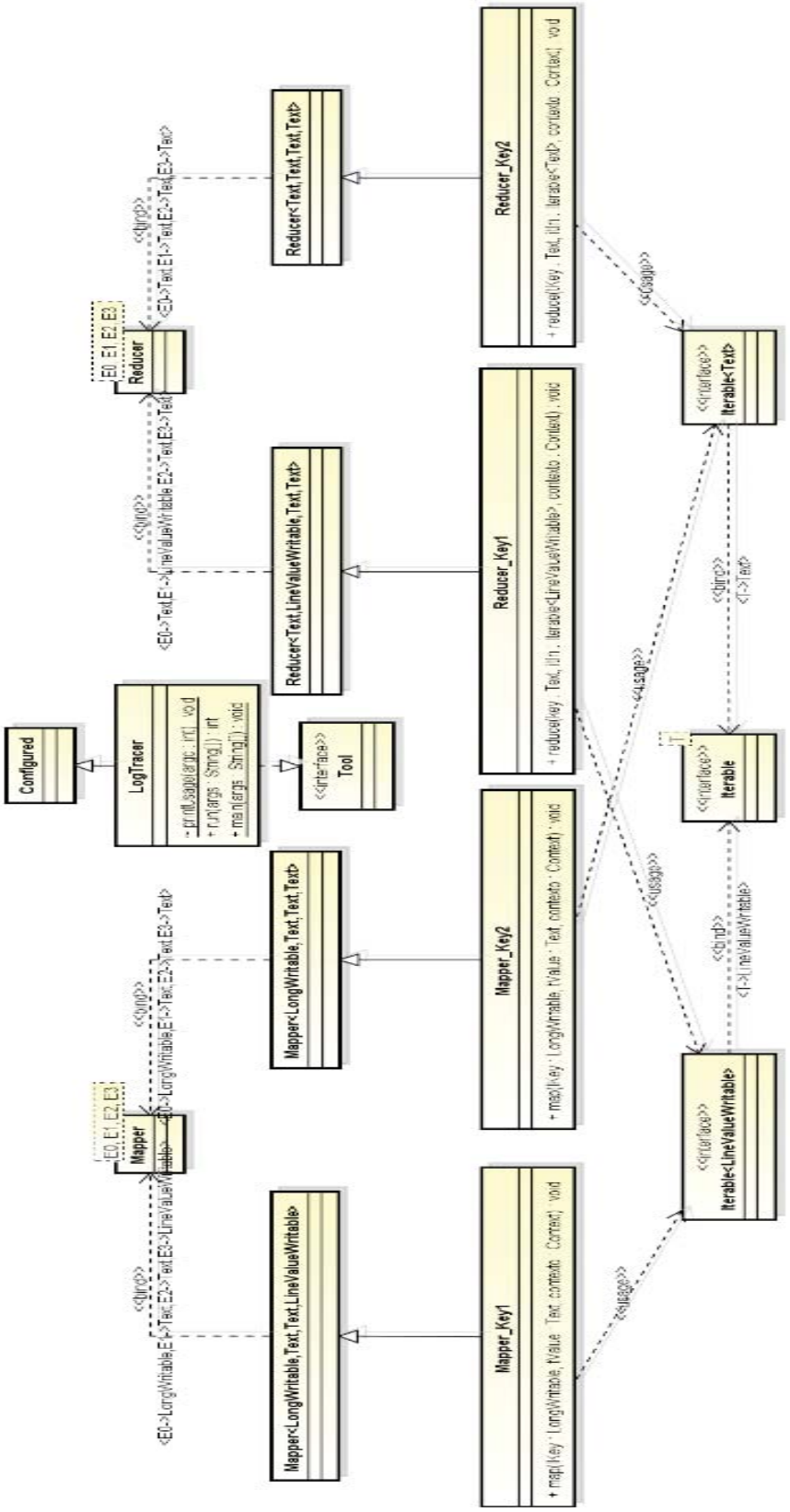

Figura 33: Diagrama UML de clases centrales del programa

Se pasa a describir las clases más estrechamente relacionadas con la secuencia MapReduce:

- Mapper: Transforma un conjunto de claves-valores a un nuevo conjunto clave-valor intermedio. Los tipos de valores de salida deben ser los mismos que los de entrada pero un par de entrada puede mapear a 0-N pares de salida. Los valores son pasados a la salida ordenados por la plataforma y asociados cada uno a su clave. Esta salida va asociada directamente a la entrada del Reducer para determinar la salida final.
- Reducer: Se encarga de realizar la lógica de cálculo final utilizando para ello pares clave-valor trasformados previamente por el Mapper. Pueden tratar desde conjuntos similares a los de la primera entrada a conjunto completamente diferentes para dar una salida que puede así mostrar facetas muy diferentes de la información que se tenía en origen. Hay una sola clave de salida tiene 3 fases primarias (Shuffle, Sort and Reduce).
- Iterable: Es un conjunto iterado estándar, pero en este caso son valores Writable (LineValueWritable/Writable) ya que los valores son varios y en este caso son de tipo “Línea” o “Texto”.
- LogTracer: Es la aplicación principal que arranca los Jobs de Hadoop. A su vez puede ser llamada por línea de comandos interpretando un path donde se encuentran los logs e implementa la interfaz Tool.
- Tool: Permite la llamada por línea de comandos facilitando así la ejecución de los Jobs de Hadoop desde un entorno distribuido (tan sólo tenemos que tener un cluster MapReduce y un NameNode que ejecute la tarea para que el resto de nodos se pongan a ejecutarlo)

6. PRESUPUESTO

A continuación se detallan los presupuestos para cada etapa del proceso de trabajo (son datos aproximados del coste que tendría la elaboración de la maqueta en caso de hacerse un presupuesto real del coste de la misma):

Tabla 7: Presupuestos parciales de la elaboración del proyecto

Etapa	Titulo	Recursos Humanos	Recursos Materiales	Otros	TOTAL
1	Búsqueda de Estado del Arte sobre el tema elegido	75horasx10€/hora (1 persona trabajando 75 horas)	0€	0€	750€
2	Análisis de infraestructura y herramientas necesarias	25horasx10€/hora (1 persona trabajando 20 horas)	0€	0€	250€
3	Coste de la infraestructura para nuestra maqueta	0€	1200€ (Equipo de sobremesa con 16GB de RAM y 8 núcleos)	0€	1200€
4	Montaje de la plataforma	75horasx10€/hora (1 persona trabajando 75 horas)	0€	0€	750€
5	Elaboración del algoritmo	50horasx10€/hora	0€	0€	500€
6	Redacción del documento	75horasx10€/hora	0€	0€	750€
7	Encuadernación del documento y presentación del mismo	0€	220€	0€	220€
	TOTAL:	**3000€**	**1420€**	**0€**	**4420€**

7. MONTAJE DE LA PLATAFORMA Y PRUEBAS DE RENDIMIENTO

7.2. MONTAJE DE LA PLATAFORMA

1. Descargar iso de CentOS 6.5 de la web: http://centos.mirror.xtratelecom.es/6.5/isos/x86_64/CentOS-6.5-x86_64-minimal.iso

2. Crear una máquina virtual vacía en VMWare Workstation. Es necesario seleccionar la opción "Custom (advanced)":

Figura 34: Configuración avanzada de creación de la máquina virtual

Es importante que en una de las pantallas en las que se da a elegir el origen desde el que instalar el sistema operativo se escoja la opción "I will install the operating system later":

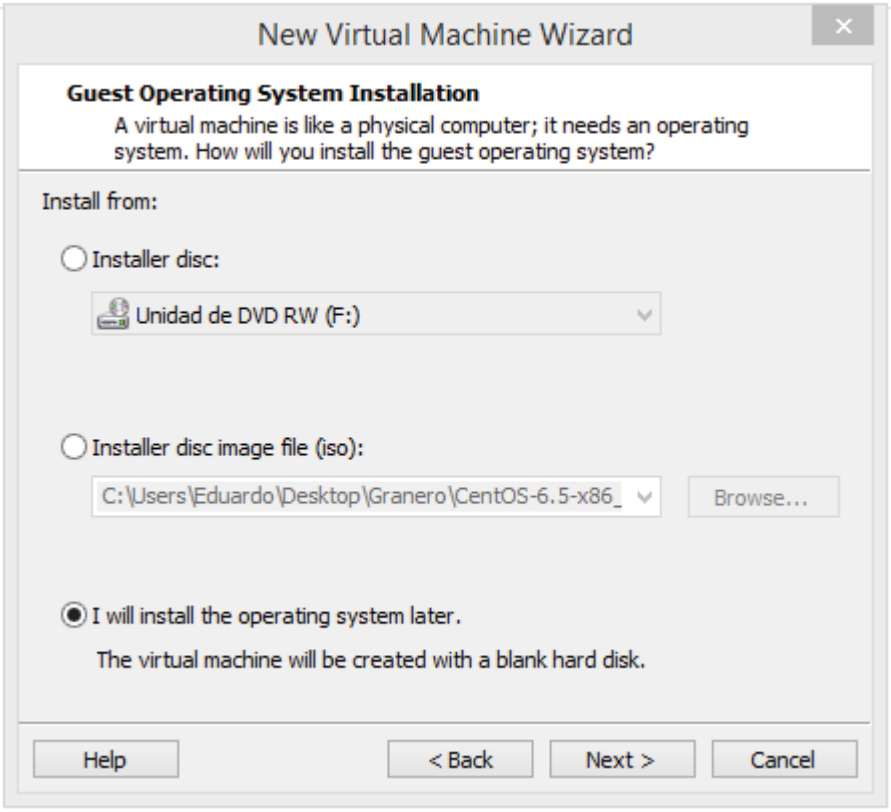

Figura 35: Opción de máquina sin sistema operativo en arranque

Escoger la opción CentOS 64-bit, 1 procesador y 1 núcleo por procesador, una memoria RAM de 1024MB, traducción NAT, SCSI Controller LSI Logic, disco duro virtual SCSI (nuevo disco virtual vacío) y disco de 40GB (con la opción "Split virtual disk into multiple files" habilitada).

Arrancar la máquina virtual introduciendo en la unidad de CD virtual la imagen .iso del sistema operativo y cargará la pantalla de instalación, eligiendo la opción de "Install or upgrade an existing system":

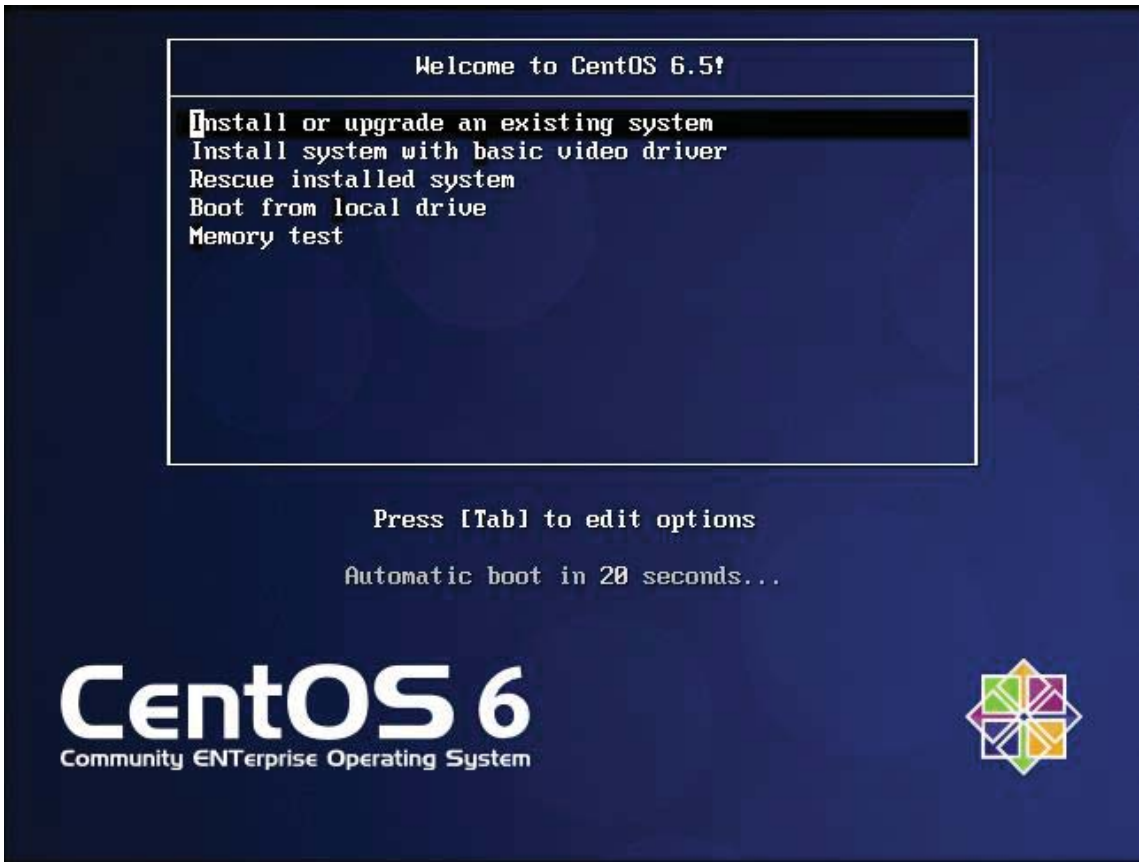

Figura 36: Pantalla de inicio del proceso de instalación del sistema

3. Configurar la otra máquina virtual con las siguientes características:

    a. Configurar teclado e idioma en español (proceso de instalación)

    b. Configurar los dispositivos de almacenamiento (proceso de instalación) y el sistema horario

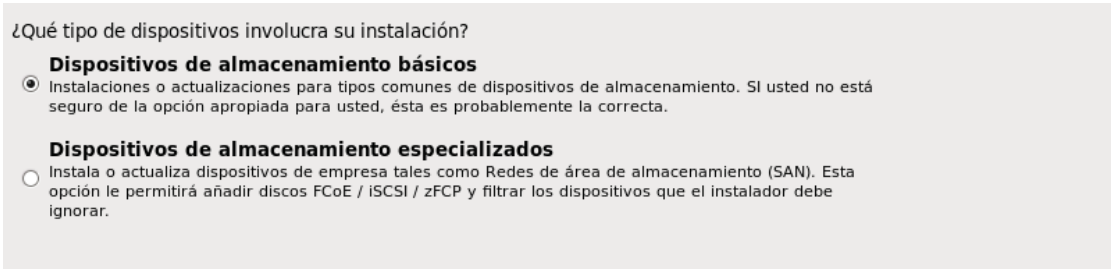

Figura 37: Configuración del tipo de almacenamiento del sistema

Se eliminarán todos los posibles datos (ninguno porque es virtual) que pudiera contener el disco.

Se fijará el horario del servidor indicando la posición en el mapa y el formato horario (UTC)

c. Se debe configurar la contraseña para el usuario root (proceso de instalación) y escoger la opción adecuada para que ocupe todo el espacio del disco:

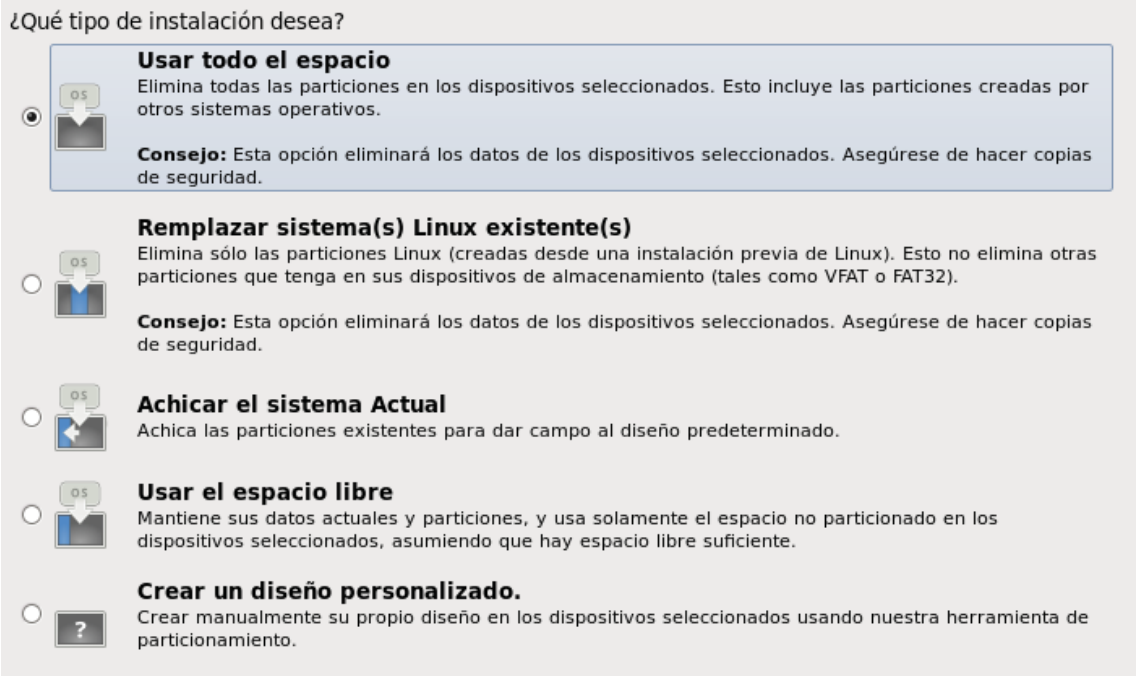

Figura 38: Configuración del sistema operativo para ocupar todo el disco disponible

d. Configurar usuario cloudera: ***adduser cloudera***

Y se configurará una password:

```
[root@clouderaclient1 ~]# passwd cloudera
Cambiando la contraseña del usuario cloudera.
Nueva contraseña:
CONTRASEÑA INCORRECTA: Está basada en una palabra del diccionario.
Vuelva a escribir la nueva contraseña:
passwd: todos los tokens de autenticación se actualizaron exitosamente.
```

Figura 39: Cambio de password para el usuario cloudera

Se comprueba que se ha creado:

```
[root@clouderaclient1 ~]# cat /etc/passwd | grep cloudera
cloudera:x:500:500::/home/cloudera:/bin/bash
```

Figura 40: Comprobación del usuario cloudera

e. Configurar usuario cloudera como sudoer en /etc/sudoers

```
[root@clouderaclient1 ~]# cat /etc/sudoers | grep cloudera
cloudera        ALL=(ALL)       NOPASSWD: ALL
```

Figura 41: Configuración como "sudoer" del usuario cloudera

f. Desactivar iptables (eliminar posibles flujos de red no permitidos por defecto):

```
[root@clouderaclient1 ~]# service iptables stop
[root@clouderaclient1 ~]# _
```

Figura 42: Detención del servicio "iptables"

g. Configurar iptables apagado desde arranque:

```
[root@clouderaclient1 ~]# chkconfig iptables off
[root@clouderaclient1 ~]# _
```

Figura 43: Eliminación del servicio "iptables" del fichero de arranque del sistema

h. Desactivar firewall (en la interfaz gráfica si la hay)

i. Si se está trabajando con la versión minimal de CentOS hay que activar la red modificando el fichero /etc/sysconfig/network-scripts/ifcfg-eth0 de manera que el parámetro "ONBOOT" valga "yes"

```
HWADDR=00:0C:29:99:69:B9
TYPE=Ethernet
UUID=bd0c6694-656b-4bed-baca-f02919911dcb
ONBOOT=yes
NM_CONTROLLED=yes
BOOTPROTO=dhcp
```

Figura 44: Activación de la red modificando el parámetro ONBOOT de la interfaz

Y reiniciar la red con "service network restart". Así se podrán instalar los paquetes o herramientas que consideremos necesarias para administrar el servidor

j. Desactivar SELinux (filtro de seguridad muy restrictivo presente en CentOS y RedHat que dificulta enormemente instalar algunas herramientas no estándar como puede ser Hadoop en este caso)

```
[root@clouderaclient1 ~]# cat /etc/sysconfig/selinux

# This file controls the state of SELinux on the system.
# SELINUX= can take one of these three values:
#     enforcing - SELinux security policy is enforced.
#     permissive - SELinux prints warnings instead of enforcing.
#     disabled - No SELinux policy is loaded.
SELINUX=disabled
# SELINUXTYPE= can take one of these two values:
#     targeted - Targeted processes are protected,
#     mls - Multi Level Security protection.
SELINUXTYPE=targeted
```

Figura 45: Desactivación del módulo de seguridad SELinux

k. Edita /etc/sysctl.conf y fija el parámetro "vm.swapiness = 0". Se hace esto porque luego se solicitará en el proceso de instalación. Es necesario que la máquina virtual no haga "swap"(balanceo) a disco para que Hadoop funcione a un mayor rendimiento:

```
[root@clouderaclient1 ~]# cat /etc/sysctl.conf | grep swapiness
vm.swapiness = 0
```

Figura 46: Desactivación del balanceo en disco de la memoria

l. Configurar /etc/hosts/

```
[root@clouderaclient1 ~]# cat /etc/hosts
127.0.0.1   localhost
192.168.58.131 clouderaclient1.apt.edu clouderaclient
```

Figura 47: Configuración del fichero /etc/hosts

m. Configurar interfaz de red (alternativa: system-config-network-tui (m))

```
[root@clouderaclient1 ~]# cat /etc/sysconfig/network-scripts/ifcfg-eth0
DEVICE=eth0
HWADDR=00:0c:29:99:69:b9
TYPE=Ethernet
UUID=551d53eb-d820-4a00-b23e-245b4b5d2b38
ONBOOT=yes
NM_CONTROLLED=yes
BOOTPROTO=none
USERCTL=no
IPV6INIT=no
IPADDR=192.168.58.131
NETMASK=255.255.255.0
DNS2=8.8.8.8
GATEWAY=192.168.58.2
DNS1=192.168.58.129
```

Figura 48: Configuración de red de modo manual editando ifcfg-eth0

n. Se instala la herramienta system-config-network-tui:

```
[root@clouderaclient1 tmp]# yum install system-config-network-tui
Loaded plugins: fastestmirror
Loading mirror speeds from cached hostfile
 * base: mirror.trueinter.net
 * extras: mirror.trueinter.net
 * updates: mirror.trueinter.net
Setting up Install Process
```

Figura 49: Instalación de la herramienta system-config-network-tui

o. Configurar nombre de host y dominio en system-config-network-tui

Figura 50: Pantalla inicial de configuración de system-config-network-tui

Figura 51: Pantalla "Configuración de DNS" de system-config-network-tui

p. Configurar IP en system-config-network-tui (alternativa al paso m)

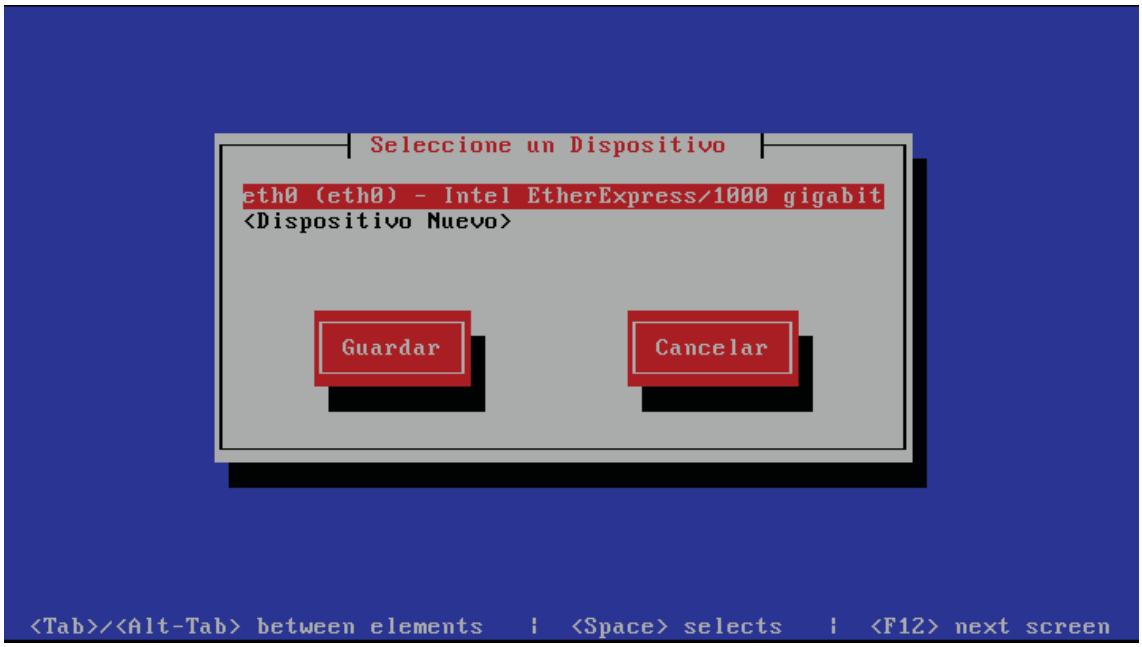

Figura 52: Selección de dispositivo a configurar con system-config-network-tui

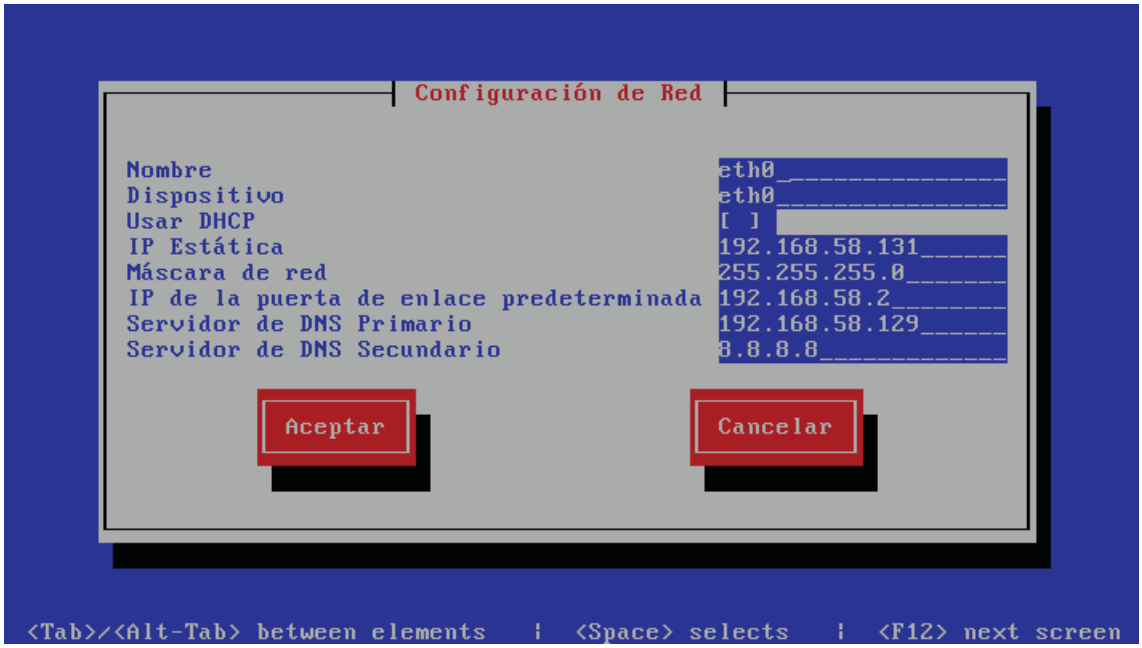

Figura 53: Configuración de interfaz de red con system-config-network-tui

q. Reiniciar la red

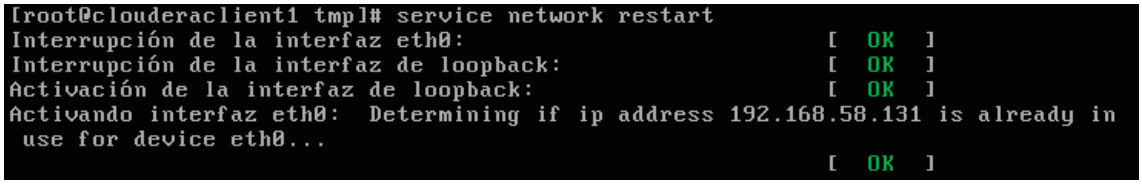

Figura 54: Proceso de reinicio de red para configurar la interfaz

r. Comprobar datos de red por interfaz GUI/CLI y ping

s. Configurar la hora por NTP correctamente a tu huso horario (el ejemplo se corresponde para CLI):

Es muy importante que se configure correctamente la hora para que se mantenga sincronizada ya que el cluster realiza tareas de heartbeat y monitorización de equipos y tiene en cuenta la hora de todas las máquinas para dar por "muerta" a una que tarda en responder

Se descarga la herramienta de sincronización horaria (NTP)

```
[root@clouderaclient1 etc]# yum install ntp
Loaded plugins: fastestmirror
Loading mirror speeds from cached hostfile
 * base: mirror.trueinter.net
 * extras: mirror.trueinter.net
 * updates: mirror.trueinter.net
Setting up Install Process
```

Figura 55: Descarga del demonio NTP

Se configuran los servidores de horario español en el fichero /etc/ntp.conf

```
[root@clouderaclient1 etc]# cat /etc/ntp.conf | grep server
# Use public servers from the pool.ntp.org project.
server 0.es.pool.ntp.org iburst
server 1.es.pool.ntp.org iburst
server 2.es.pool.ntp.org iburst
server 3.es.pool.ntp.org iburst
#broadcast 192.168.1.255 autokey        # broadcast server
#broadcast 224.0.1.1 autokey            # multicast server
#manycastserver 239.255.254.254         # manycast server
```

Figura 56: Configuración de servidores españoles para NTP

Se "matan" todos los procesos de NTP que existan hasta el momento:

```
[root@clouderaclient1 etc]# killall ntpd
[root@clouderaclient1 etc]#
```

Figura 57: Eliminación de procesos NTP existentes

Se fuerza la sincronización de la hora local con uno de los servidores NTP locales:

```
[root@clouderaclient1 etc]# ntpdate 0.es.pool.ntp.org
 2 Jun 16:52:39 ntpdate[59149]: step time server 81.19.96.148 offset 41502.48695
5 sec
```

Figura 58: Proceso de sincronización forzada de la hora con servidor NTP español

Se reinicia la sincronización de la hora de manera automática levantando el servicio NTP:

```
[root@clouderaclient1 etc]# service ntpd start
Iniciando ntpd:                                            [  OK  ]
```

Figura 59: Reinicio del servicio NTP

Se comprueba que la hora es la correcta:

```
[root@clouderaclient1 etc]# date
lun jun  2 16:53:46 CEST 2014
```

Figura 60: Obtención de la fecha actual sincronizada

t. Reiniciar la máquina y volver a comprobarlo todo

4. Clonar la máquina virtual y configurar un servidor de DNS con las siguiente zona:

```
$ORIGIN .
$TTL 3600; 1 hora
apt.edu          IN SOA          ns1.apt.edu. root.apt.edu. (
                                                 1; serial
                                                 1D; refresh
                                                 1H; retry
                                                 1W; expire
                                                 3H); minimum
                                        NS       192.168.58.129
                                        A        192.168.58.130
$ORIGIN apt.edu
ns1              A               192.168.58.129
cmanager         A               192.168.58.130
clouderaclient1  A               192.168.58.131
clouderaclient2  A               192.168.58.132
www              CNAME           cmanager
```

Figura 61: Zona DNS apt.edu con la resolución del cluster

Se podría haber configurar la zona en el fichero /etc/hosts de cada máquina, lo que evitaría instalar un servicio DNS, pero se ha preferido instalarlo debido 2 razones:

1. Con un servicio DNS se podrán hacer pruebas en un futuro usando el servidor DNS como beta-testing de la propia herramienta de detección malware.
2. Con el servidor DNS podremos acceder a la máquina cmanager.apt.edu desde el host anfitrión fijando como servidor DNS global la IP 192.168.58.129 y a través del nombre www.apt.edu (Alias de cmanager.apt.edu).

Se procede a instalar el servicio DNS con el comando: "***yum install bind***". Luego se ejecuta "***service named start***" para arrancar el servicio y "***chkconfig named on***" para que se inicie con el arranque del sistema.

Se edita el fichero /etc/named.rfc1912.zones y se añade una referencia a la nueva zona:

```
zone "apt.edu" {
        type master;
        file "db.apt.edu";
        allow-update { none; };
};
```

Figura 62: Zona apt.edu definida el servidor DNS

Se crea el fichero **/var/named/db.apt.edu** introduciendo el contenido de zona anteriormente indicado. Se edita el fichero **/etc/named.conf,** concretamente borrando la línea **listen-on port 53** (para que por defecto el servidor funcione por todas las interfaces), añadiendo la configuración de **forwarders** (se redirige al servidor público 8.8.8.8 todo lo aquello de lo que no se tenga conocimiento) y permitiendo a cualquier host preguntar al DN**S** (**editar allow-query** sustituyendo "**localhost**" por "**any**"):

```
options{
        forwarders{
                8.8.8.8;
                8.8.4.4;
        };
        listen-on-v6 port 53 { ::1; };
        directory       "/var/named";
        dump-file       "/var/named/data/cache_dump.db";
        statistics-file "/var/named/data/named_stats.txt";
        memstatistics-file "/var/named/data/named_mem_stats.txt";
        allow-query     { any; };
```

Figura 63: Se editan las características de /etc/named.conf

El servicio se reinicia(service named restart) y se prueba que funciona realizando una resolución DNS:

```
[root@clouderaclient1 named]# service named restart
Deteniendo named:                                          [  OK  ]
Iniciando named:                                           [  OK  ]
[root@clouderaclient1 named]# host www.apt.edu
www.apt.edu is an alias for cmanager.apt.edu.
cmanager.apt.edu has address 192.168.58.130
```

Figura 64: Resolución DNS del cluster funcionando

Se seguirá el mismo proceso realizado con la zona apt.edu para la zona inversa (58.168.192.in-addr.arpa):

```
[root@ns1 named]# cat db.192.168.58
$ORIGIN .
$TTL 3600; 1 hora
58.168.192.in-addr.arpa         IN SOA          ns1.apt.edu. root.apt.edu. (
                                                1; serial
                                                1D; refresh
                                                1H; retry
                                                1W; expire
                                                3h); minimum
                                        NS      192.168.58.129
$ORIGIN 58.168.192.in-addr.arpa
129             PTR             ns1.apt.edu.
130             PTR             cmanager.apt.edu.
131             PTR             clouderaclient1.apt.edu.
132             PTR             clouderaclient2.apt.edu.
```

Figura 65: Zona inversa / reversal zone de los hosts del cluster

Además, se puede cambiar el DNS del host anfitrión para poder resolver también los nombres publicados en el cluster:

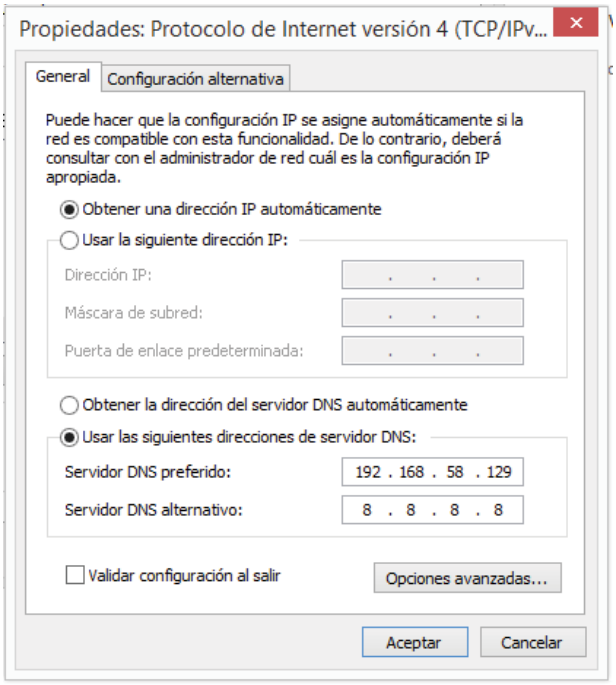

Figura 66: Se configura el servidor DNS también en el host anfitrión

Comprobamos que desde el host anfitrión también se resuelve correctamente:

```
C:\Users\Eduardo>nslookup www.apt.edu
Servidor:  UnKnown
Address:  192.168.58.129

Nombre:  cmanager.apt.edu
Address:  192.168.58.130
Aliases:  www.apt.edu
```

Figura 67: Resolución de nombre de host del cluster desde el host anfitrión

5. Clonar la máquina virtual (sin servidor de DNS) hasta conseguir otras 4 máquinas virtuales en total:

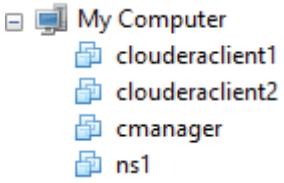

Figura 68: Máquinas virtuales clonadas a partir de clouderaclient1

6. Cambiar lo necesario para cada máquina virtual

   Es imprescindible cambiar la IP por la correspondiente en cada caso, el nombre en el fichero **/etc/hosts**, el hostname con **system-config-network-tui** y muy probablemente sea necesaria cambiar la MAC del interfaz de red. Se puede averiguar cuál es la nueva interfaz (en ocasiones se renombra) y cuál es la nueva MAC con el comando ifconfig –a:

```
[root@clouderaclient1 ~]# ifconfig -a
eth1      Link encap:Ethernet  HWaddr 00:0C:29:65:5C:78
          inet addr:192.168.58.132  Bcast:192.168.58.255  Mask:255.255.255.0
          inet6 addr: fe80::20c:29ff:fe65:5c78/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:46 errors:0 dropped:0 overruns:0 frame:0
          TX packets:58 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:3047 (2.9 KiB)  TX bytes:2652 (2.5 KiB)
```

Figura 69: Se ha renombrado la interfaz y cambiado la MAC al clonar

Se procede a copiar la dirección MAC (HWaddr) y se edita manualmente el fichero /etc/sysconfig/network-scripts/ifcfg-eth0, que también queda renombrado a "ifcfg-eth1":

```
[root@clouderaclient1 ~]# cat /etc/sysconfig/network-scripts/ifcfg-eth1
DEVICE=eth1
HWADDR=00:0c:29:65:5c:78
TYPE=Ethernet
UUID=bd0c6694-656b-4bed-baca-f02919911dcb
ONBOOT=yes
NM_CONTROLLED=yes
BOOTPROTO=none
USERCTL=no
IPV6INIT=no
IPADDR=192.168.58.132
NETMASK=255.255.255.0
GATEWAY=192.168.58.2
DNS1=192.168.58.129
DNS2=8.8.8.8
```

Figura 70: Se cambia el fichero de interfaz a la nueva MAC y nombre

Se reinicia la red para que los cambios se vean reflejados en la interfaz.

7. Generar la clave ssh de cada máquina y exportar la clave de la máquina manager a las clientes

```
[root@cmanager ~]# ssh-keygen
Generating public/private rsa key pair.
Enter file in which to save the key (/root/.ssh/id_rsa):
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /root/.ssh/id_rsa.
Your public key has been saved in /root/.ssh/id_rsa.pub.
The key fingerprint is:
80:f5:94:8a:bc:ab:2d:aa:85:7a:23:d3:de:f3:ce:74 root@cmanager.apt.edu
The key's randomart image is:
+--[ RSA 2048]----+
¦       . ..      ¦
¦      o o.       ¦
¦    ......       ¦
¦     o ..        ¦
¦       .  S       ¦
¦ .  .             ¦
¦...  .. E         ¦
¦+.=ooo .          ¦
¦=*o+o++           ¦
+-----------------+
```

Figura 71: Generación de la clave pública SSH

```
ssh-copy-id clouderaclient2
root@clouderaclient2's password:
Now try logging into the machine, with "ssh 'clouderaclient2'", and check in:

  .ssh/authorized_keys

to make sure we haven't added extra keys that you weren't expecting.
```

Figura 72: Exportar clave pública de cmanager al resto de máquinas

8. Instalar en una de ellas (cmanager) Cloudera Manager e instalar el cluster al completo

    a. Descarga en la máquina cmanager del software "Cloudera Manager" de la dirección http://archive.cloudera.com/cm5/installer/latest/cloudera-manager-installer.bin Ejecutando:

    cd /home/cloudera/Desktop

    wget http://archive.cloudera.com/cm5/installer/latest/cloudera-manager-installer.bin

    b. Cambiar los permisos al fichero (desde root):

    chmod 750 cloudera-manager-installer.bin

c. Ejecutarlo (./cloudera-manager-installer.bin) y seguir los pasos

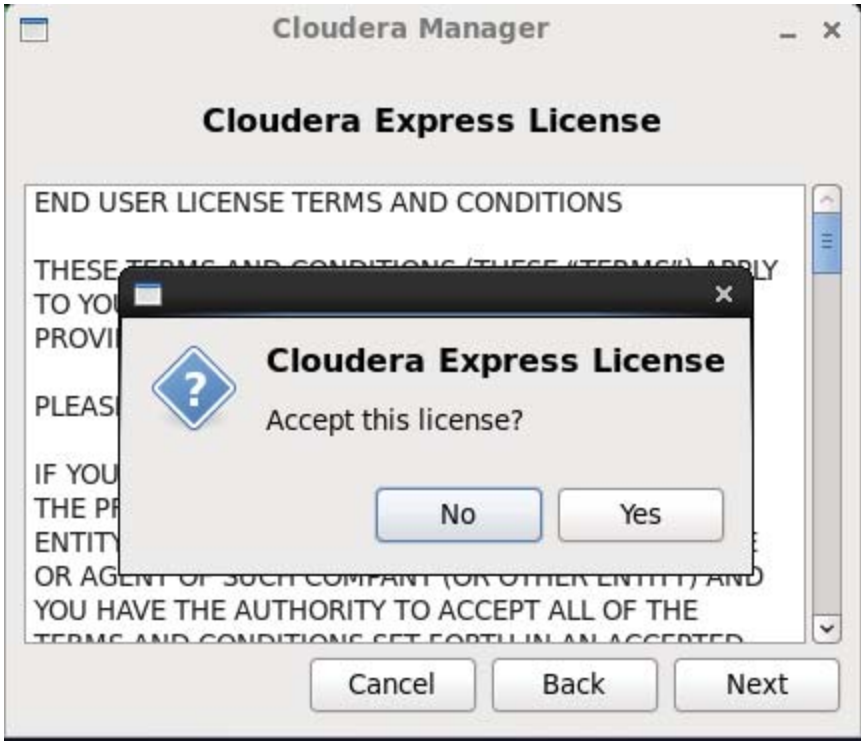

Figura 73: Captura de pantalla de la aceptación de la licencia para la instalación

d. Abrir, tal y cómo se indica, el navegador en la página localhost:7180:

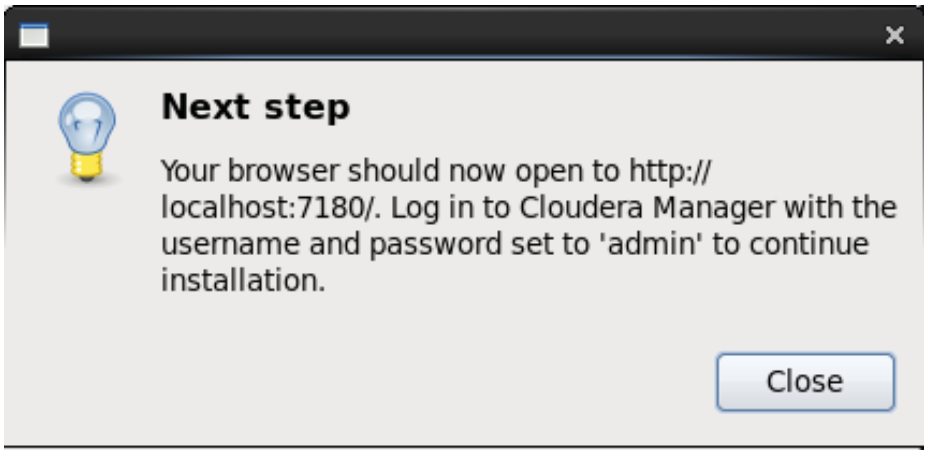

Figura 74: Pantalla del fin del proceso de instalación de Cloudera Manager

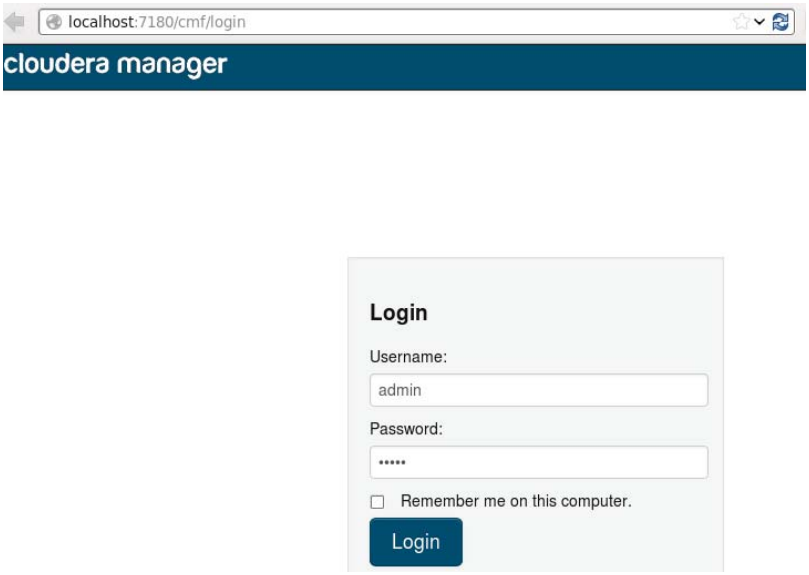

Figura 75: Captura de pantalla del login en la interfaz web de Cloudera Manager

e. Hacer que el servicio cloudera-scm-server arranque al iniciar la máquina (desde otro terminal):

   chkconfig cloudera-scm-server on

f. Hacer log in y añadir los nodos con los que queremos formar el cluster (sus nombres), proceder a la instalación en remoto del software en los hosts:

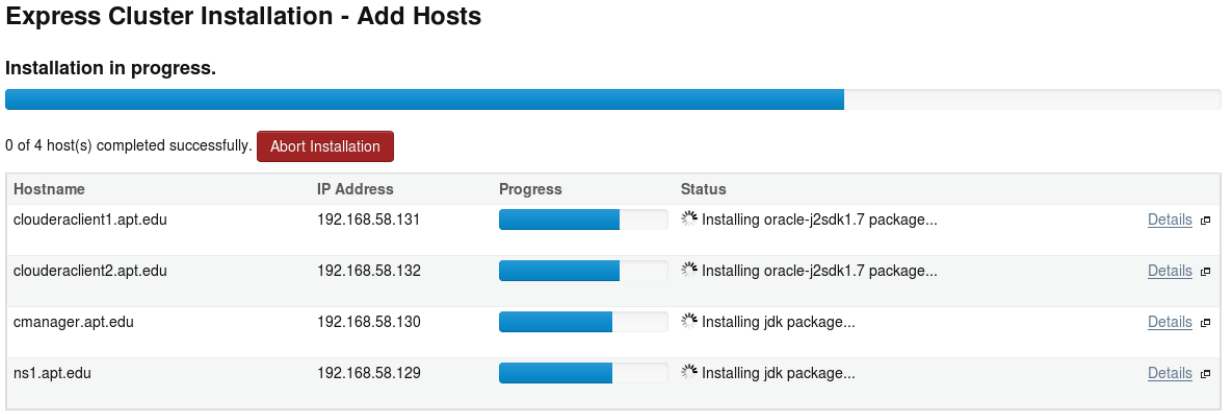

Figura 76: Pantalla del progreso del proceso de distribución de la plataforma

g. Descargar desde cloudera manager el CDH 4.6 y distribuirlo al resto de equipos

h. Configurar el role que va a tomar cada nodo del cluster. El nodo central en este caso será el nodo cmanager, sobre el que caerán las funciones de ordenar las tareas y monitorizar y mostrar al usuario no sólo el estado del cluster (Cloudera Manager), sino también trabajos y reportes. Para clarificar la repartición de las tareas y roles de los hosts del cluster se ha

expuesto en las siguientes capturas de pantalla cuales son los definitivos roles que se han configurado en cada uno:

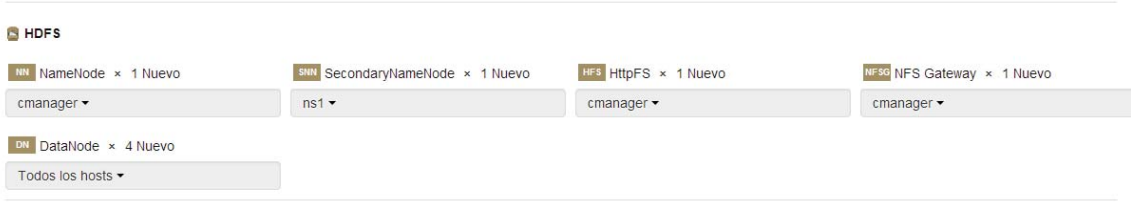

Figura 77: Captura de repartición de roles HDFS del cluster

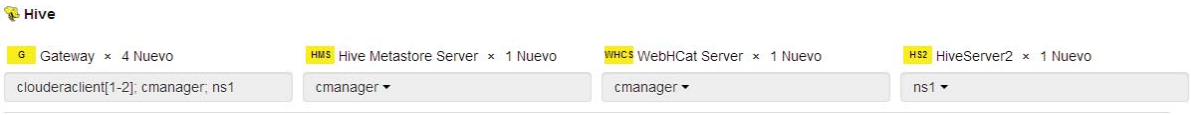

Figura 78: Captura de repartición de roles Hive del cluster

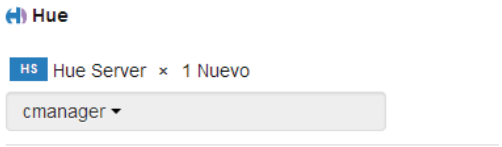

Figura 79: Captura de repartición de roles Hue del cluster

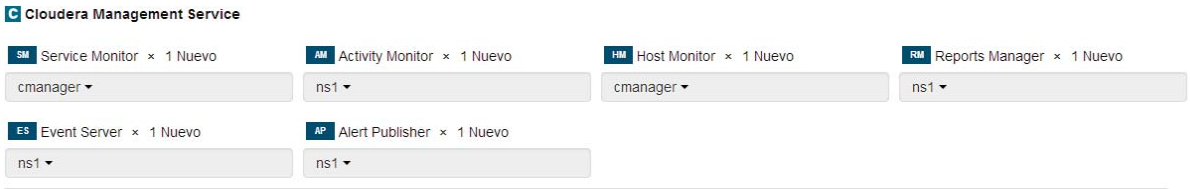

Figura 80: Captura de repartición de roles CMS del cluster

Figura 81: Oozie server del cluster

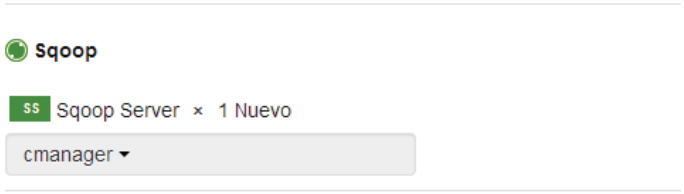

Figura 82: Sqoop server del cluster

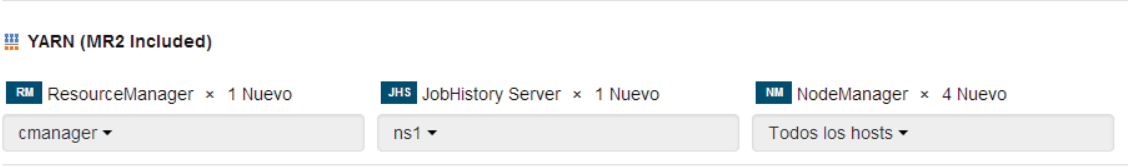

Figura 83: Repartición de roles YARN/MapReduce 2 del cluster

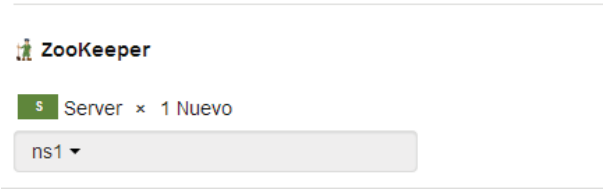

Figura 84: Zookeper Server del cluster

i. Configurar las bases de datos a usar para cada funcionalidad del cluster:

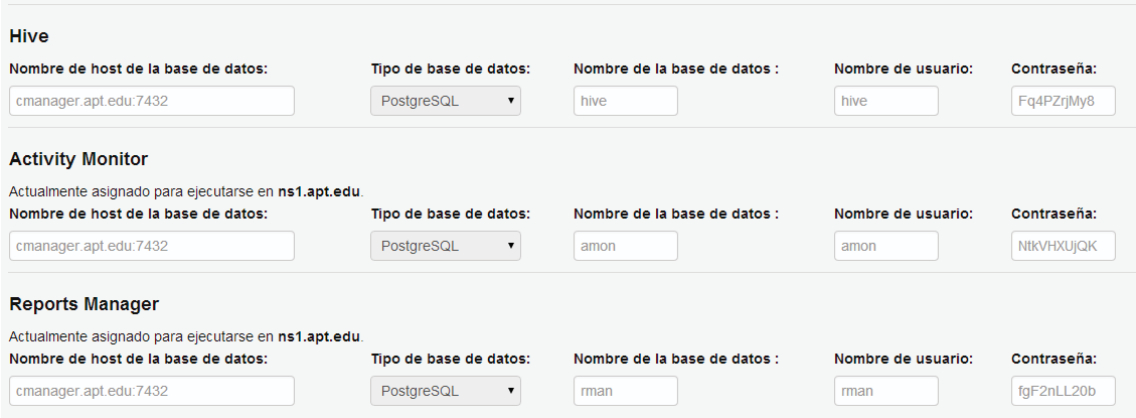

Figura 85: Pantalla de configuración de las bases de datos de la plataforma

j. Ejecutar todas las acciones y esperara a que se instalen todos los servicios

7.3. PRUEBAS DE RENDIMIENTO

Una vez configurado el cluster se ha pasado a probar que las máquinas pueden dar un rendimiento adecuado para la configuración de los servicios y la ejecución de los Jobs necesarios para hacer funcionar el proyecto.

Lo primero que se puede observar con la configuración antes mencionada es que **el servidor virtualizado cmanager.apt.edu no puede arrancar correctamente el servicio "cloudera-scm-server"** tras poner la variable "vm.swappiness = 0". Eliminamos el "tunning" de kernel y devolvemos a su valor por defecto dicha variable y probamos a observar cual es el factor de SWAP en disco (en principio no debería haber SWAP si tenemos en cuenta que la máquina tiene una gran cantidad de memoria RAM) y observamos este resultado haciendo un comando top:

```
top - 20:48:27 up  1:22,  1 user,  load average: 3.66, 3.17, 3.01
Tasks: 138 total,   1 running, 137 sleeping,   0 stopped,   0 zombie
Cpu0  : 10.9%us,  2.5%sy,  0.0%ni, 54.7%id, 31.2%wa,  0.4%hi,  0.4%si,  0.0%st
Mem:   2948724k total,  2879692k used,    69032k free,     3324k buffers
Swap:  4095992k total,   279200k used,  3816792k free,    61364k cached

  PID USER      PR  NI  VIRT  RES  SHR S %CPU %MEM    TIME+  COMMAND
 3606 cloudera  20   0 3282m 669m 6500 S 10.6 23.2   3:39.37 java
 7257 cloudera  20   0 2119m 353m  13m S  3.3 12.3   1:54.38 java
 1224 root      20   0 1105m  43m 2376 S  2.0  1.5   0:32.25 python
 9019 yarn      20   0  800m 136m 6584 S  1.0  4.7   0:50.04 java
15645 root      20   0 1243m  15m 7088 S  1.0  0.5   0:00.03 java
 3624 cloudera  20   0  658m  10m 5508 S  0.7  0.4   0:02.14 postgres
 1288 root      20   0  198m 7888 1504 S  0.3  0.3   0:07.53 python
 7239 cloudera  20   0 2007m 241m 8884 S  0.3  8.4   0:50.75 java
 8325 hdfs      20   0 1693m 221m 6540 S  0.3  7.7   0:23.10 java
 8340 hdfs      20   0  634m 105m 6496 S  0.3  3.6   0:15.96 java
 8487 hdfs      20   0  559m  76m 6032 S  0.3  2.7   0:05.27 jsvc
 8770 yarn      20   0  665m 113m 6528 S  0.3  3.9   0:16.80 java
 9393 hive      20   0  618m 106m 6292 S  0.3  3.7   0:07.26 java
 9718 oozie     20   0  897m 133m 6364 S  0.3  4.6   0:41.84 java
 9908 sqoop2    20   0  869m 105m 6260 S  0.3  3.6   0:07.82 java
15422 root      20   0 15028 1332 1004 R  0.3  0.0   0:01.07 top
15642 root      20   0  105m 1604 1320 S  0.3  0.1   0:00.01 java_version.sh
    1 root      20   0 19232  976  844 S  0.0  0.0   0:00.68 init
```

Figura 86: Consumo de memoria (enorme) realizado por los procesos del sistema

Tal y cómo se puede mostrar en la imagen tan sólo los procesos relacionados con el usuario cloudera ya consumen por sí solos en algunos momentos más de 5Gb de RAM, por lo que la máquina se ve obligada a realizar SWAP para poder funcionar. Esto supone que la máquina se encuentra totalmente desbordada ante los requerimientos de RAM de nuestra plataforma distribuida.

También es posible obtener conclusiones a partir de la propia información que nos da Cloudera Manager. El primer dato relativo a rendimiento que se obtiene de la plataforma está directamente relacionado con gráficas de Lectura/Escritura, uso de red y consumo de memoria y CPU. Tal y cómo se observa, los resultados son los siguientes:

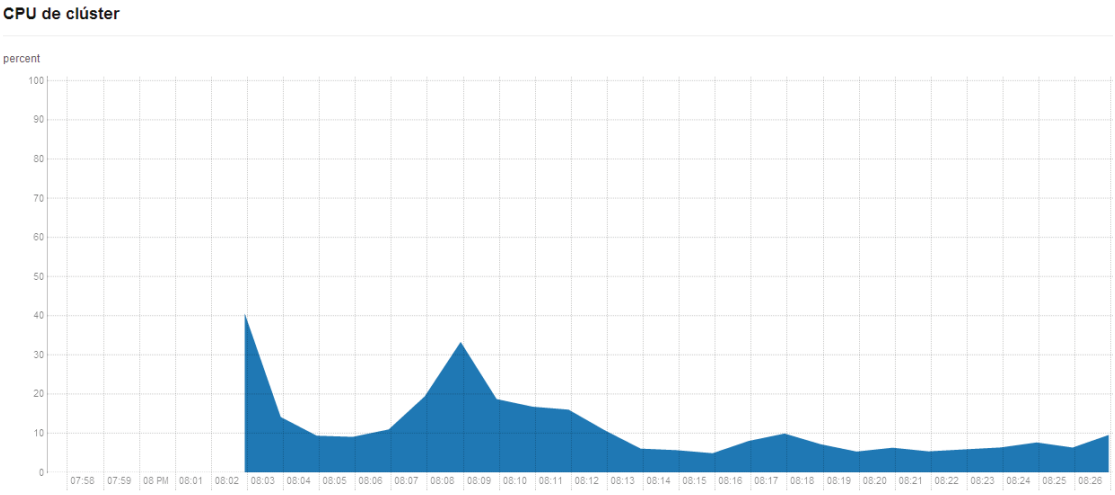

Figura 86: Fluctuación de CPU injustificada en el cluster

Esta gráfica indica que la CPU no está sobrecargada, pero sí que hay picos de CPU (a pesar de que el sistema no está realizando ningún tipo de cálculo), lo que puede dejar en evidencia que el sistema puede estar realizando tareas imprevistas (como escribir en disco). Es por esto por lo que se pasa a observar la gráfica de E/S de datos en el cluster:

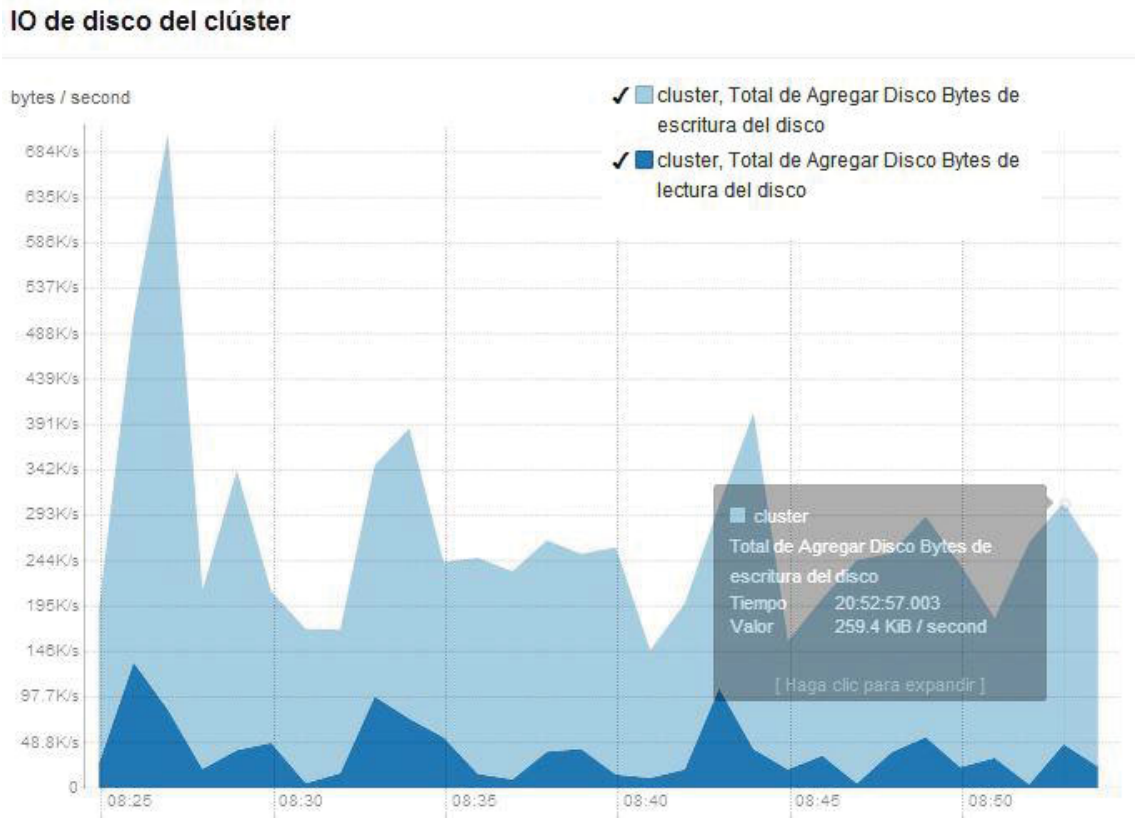

Figura 88: Picos grandes de operaciones E/S con el cluster en reposo

Si se observa la gráfica superior se puede fácilmente confirmar la suposición anterior. El sistema, a pesar de encontrarse en absoluto reposo está realizando una tarea

continua de Lectura/Escritura en disco y esto es observable en los picos de IO que se detectan.

En el panel de control del cluster se observan errores relativos a los nodos que lo componen. Estos errores corroboran lo que inicialmente se ha considerado; el consumo mínimo de RAM del equipo para su buen funcionamiento es superior del que actualmente disponemos:

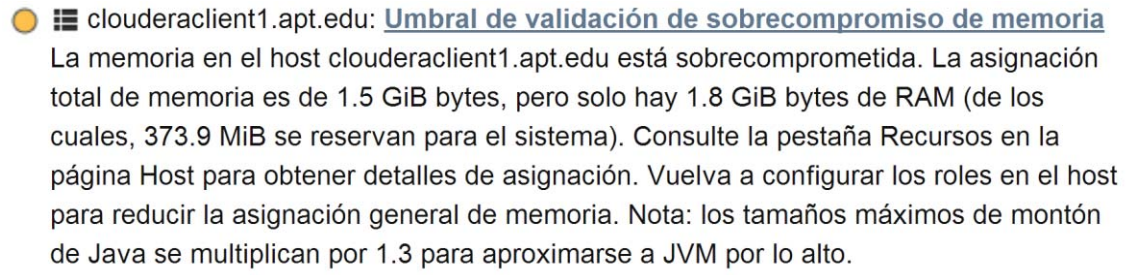

Figura 89: Memoria insuficiente en clouderaclient1

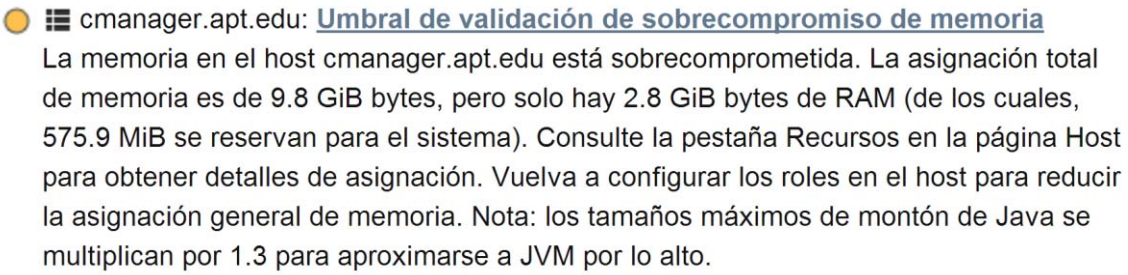

Figura 90: Memoria insuficiente en cmanager

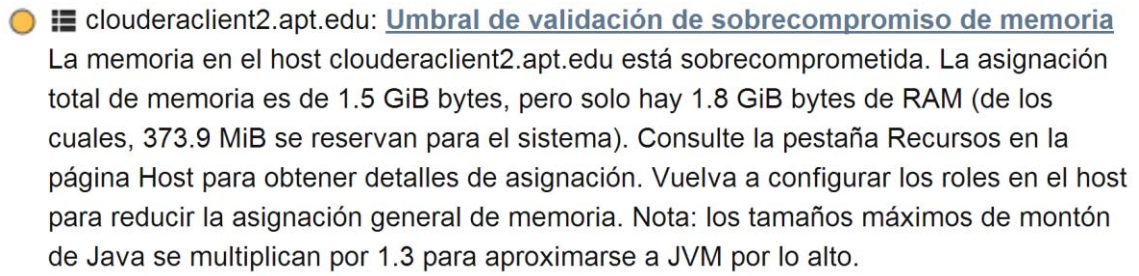

Figura 91. Memoria insuficiente en clouderaclient2

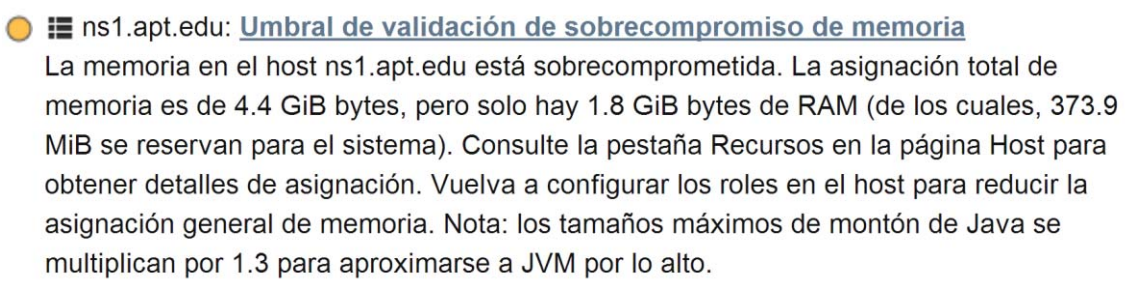

Figura 92: Memoria insuficiente en ns1

Además, de todos estos errores y en un intento de depuración de posibles fallos inesperados en el hardware o de borrar procesos con alto consumo de memoria se ha revisado el consumo de memoria total del equipo, arrojando estos esclarecedores resultados:

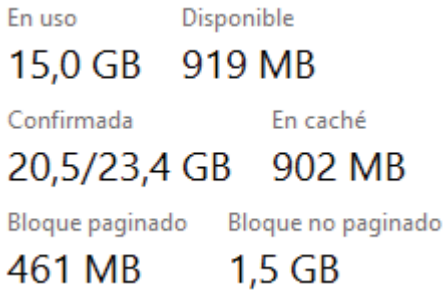

Figura 93: Consumo de memoria RAM del host anfitrión

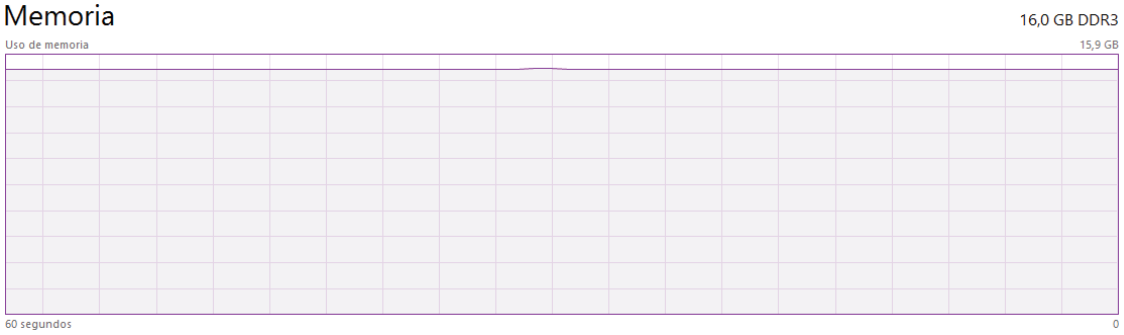

Figura 94: Gráfica de consumo de memoria del sistema anfitrión

El sistema se encuentra totalmente desbordado. El consulo de RAM de los procesos lanzados es demasiado para poder realizar ningún tipo de cálculo adicional en la plataforma. De hecho, se observa lentitud y bloqueo intermitente de procesos en el sistema anfitrión mientras se tiene encendido el cluster al completo.

En esta situación se puede considerar que el cluster y la infraestructura virtualizada no sirven para realizar la tarea que en este proyecto se ha propuesto ya que la tarea requiere de unas especifaciones muy superiores con respecto a RAM. Al no disponer de un hardware más potente esta situación impide finalmente poder realizar una implementación del algoritmo propuesto con una demostración práctica.

NOTA: Tal y cómo se comenta en el apartado “Conclusiones y trabajo futuro” se observa que a partir del 10 de Junio de 2014 hay disponible una tecnología denominada OpenPOWER potenciada por la compañía tecnológica IBM que podría servir de solución a la limitación hardware aquí mostrada.

8. CONCLUSIONES Y TRABAJO FUTURO

Uno de los objetivos del proyecto ha sido investigar y experimentar algoritmos que pudieran afrontar la detección de ataques APTs desde un punto de vista distribuido basado en el análisis de logs de DNS de una compañía.

Hemos dedicado gran parte del tiempo investigando y experimentando con un algoritmo transformable al modelo MapReduce y capaz de obtener unos resultados definitivos y fiables acerca de los posibles hosts infectados dentro de una red corporativa y finalmente este algoritmo no ha podido ser implementado.

El motivo por el cual no se ha podido implementar este algoritmo es que, tal y cómo hemos visto previamente, se debe pasar un proceso de disgregar las tareas del algoritmo en subtareas traducibles a MapReduce. Esto supone reducir toda la lógica a un sistema clave-valor previamente a la inferencia final. Además, supone un impedimento con respecto a nuestra infraestructura, ya que se deberán analizar los logs DNS tantas veces como subalgoritmos procesemos y en nuestro caso esto alargaría la tarea a un tiempo de procesado que superaría las 20 horas.

Además de todo esto se puede observar en base a los datos y gráficas de rendimiento recopiladas que el cluster montado no tenía capacidad suficiente de cálculo al estar enormemente limitado su RAM (a pesar, incluso, que el sistema fue montado con los requerimientos mínimos de memoria).

El desarrollo y codificación completa del algoritmo capaz de detectar malware en base a los logs de DNS es una tarea compleja que requeriría mayor tiempo de investigación . Es por esto, por lo que hemos reenfocado el proyecto a demostrar que el proceso de Belief Propagation es realmente un algoritmo transformable al paradigma MapReduce.

En los últimos días de la elaboración del proyecto se ha abierto una nueva fuente de hardware que probablemente sea de una enorme utilidad para la investigación en este campo. Esta fuente novedosa consiste en que la empresa tecnológica y de soporte y mantenimiento hardware IBM ha lanzado un proyecto denominado “OpenPOWER Foundation” que consiste en que a partir del 10 de Junio (2014) habrá disponible un gran número indeterminado de racks con equipos de última generación IBM POWER8 (servidores) disponibles al público para poder realizar tareas de investigación relacionadas con la programación distribuida y el BigData. (IBM, 2014)[20]

Esto facilitará enormemente la tarea a posibles futuros alumnos que quieran continuar desarrollando una investigación enfocada al descubrimiento de Malware APT basado en logs de DNS usando este proyecto, ya que no tendrán la limitación de recursos de cálculo que puede tener una red doméstica.

En definitiva, se han abierto nuevas vías de investigación a partir de este proyecto y se ofrece una base de ayuda importante a partir del tutorial paso a paso de montaje de la plataforma, un diagrama UML completo (clases, secuencia, distribución, componente, casos de uso…), un pseudocódigo de implementación y se ha eliminado la limitación hardware que inicialmente teníamos.

Toda estas facilidades y la existencia de algoritmos propietarios ya funcionando nos lleva a deducir que la implementación de un sistema de detección APT y malware en una red corporativa en base a los logs DNS no está tan lejos de alcanzar una viabilidad comercial cómo a priori se pudiera esperar del problema.

Probablemente, este sistema que aquí se presenta y su arquitectura completa serían mucho más atractivos para un ámbito comercial si se pudiera proveer a los clientes de un API o interfaz de usuario que facilite la carga de datos, el cálculo de probabilidades de infección y la obtención de una lista de nodos con su probabilidad de infección.

Por todo esto, una posible ampliación de este proyecto sea desarrollar el resto de subpartes de este algoritmo en base a las premisas especificadas previamente en los anteriores apartados y dotar a la plataforma de un carácter ubicuo y comercial. Esto se podría elaborar desarrollando una API o una GUI Web y convirtiendo a las herramientas que se ofrecen en un servicio “en la nube”, probablemente en un servicio SaaS.

9. REFERENCIAS

[1] Dell Secureworks. (2012). *Secureworks Persisten Threats: Healthcare Under Attack*. [http://www.secureworks.com/assets/pdf-store/articles/Advanced_Persistent_Threats_wp_-_Healthcare.pdf] consultado en Febrero 2015

[2] Kaspersky. (2013). *Karspersky lab: Clasificaciones de malware*. [http://www.kaspersky.es/internet-security-center/threats/malware-classifications] consultado en Marzo 2014

[3] BitDefender. (2010). *Malware and Spam Survey finds e-threats adapting to online behavioral trends*. [http://www.bitdefender.com/news/bitdefender-malware-and-spam-survey-finds-e-threats-adapting-to-online-behavioral-trends-1094.html] consultado en Febrero 2014

[4] Mandiant. (2013). *Mandiant APT1 Report*. [http://intelreport.mandiant.com/Mandiant_APT1_Report.pdf] consultado en Abril 2014

[5] Roolvink, S. (15 de Diciembre de 2008). *Detecting attacks involving DNS servers*. [http://essay.utwente.nl/58497/1/scriptie_S_Roolvink.pdf] consultado en Marzo 2014

[6] IETF. (Noviembre de 1987). *RFC 1035: DOMAIN NAMES - IMPLEMENTATION AND SPECIFICATION* [http://www.ietf.org/rfc/rfc1035.txt] consultado en Mayo 2014

[7] HoneyNet. (13 de Julio de 2007). *Fast-Flux Service Networks*. [http://old.honeynet.org/papers/ff/fast-flux.html] consultado en Mayo 2014

[8] INTECO. (Mayo de 2013). *Detección de APTs*. [http://cert.inteco.es/extfrontinteco/img/File/intecocert/EstudiosInformes/deteccion_apt.pdf] consultado en Junio 2014

[9] NASA, LANL. (25 de Febrero de 2014). *APT Fingerprints in DNS Data*. [http://cps-vo.org/node/11450] consultado en Marzo 2014

[10] NASA, LANL. (30 de Abril de 2013). *APT INFECTION DISCOVERY USING DNS DATA*. [http://permalink.lanl.gov/object/tr?what=info:lanl-repo/lareport/LA-UR-13-23109] consultado en Abril 2014

[11] Apache. (s.f.). What is Apache Hadoop? [http://hadoop.apache.org/#What+Is+Apache+Hadoop%3F] consultado en Junio 2014]

[12] CentOS. (s.f.). *CentOS Project*. [https://www.centos.org/about/] consultado en Marzo 2014

[13] Cloudera. (s.f.). *Cloudera Express, the best way to get started with Hadoop*. [http://www.cloudera.com/content/cloudera/en/products-and-services/cloudera-express.html] consultado en Marzo 2014

[14]Thórisson, K. R. (s.f.). *Cognitive Architecture & Sentient Systems*. [http://www.ru.is/kennarar/thorisson/] consultado en Junio 2014

[15] Cantarero Dávila, C., & Ramírez Fernández, E. (2013). MEBN: Smart Security at Neighbourhood, Julio de 2013

[16] Zhang, X. C. (7 de Mayo de 2013). *A Simple Example to Demonstrate how does the MapReduce work*. [http://xiaochongzhang.me/blog/?p=338] consultado en Junio 2014

[17] Hadoop. (2008). *HDFS ARCHITECTURE DESIGN*.
[http://hadoop.apache.org/docs/r1.2.1/hdfs_design.html] consultado en Febrero 2014

[18] Shankar, V. (19 de Septiembre de 2011). *Learncomputer: Hadoop with hive*.
[http://www.learncomputer.com/hadoop-with-hive/] consultado en Abril 2014

[19] BDI Systems. (s.f.). *BDISYSTEMS: Big Data Hadoop*
[http://www.bdisys.com/27/1/17/BIG%20DATA/HADOOP] consultado en Mayo 2014

[20]IBM. (11 de Junio de 2014). *IBM ships Power Systems servers targeting Big Data opportunities* [www-03.ibm.com/press/] consultado el 11 de Junio de 2015